

# **Task Force Web Developer's Guidance**

***TF*** **Web**

November 9, 2001, v 2.0

DEV400-U-A57006-APPL-TFWEBdevelopersguidance-v2.0

## Change History

The following Change History log contains a record of changes made to this document:

| Published / Revised Date | Version # | Author (optional)    | Section / Nature of Change   |
|--------------------------|-----------|----------------------|--|
| 07/03/2001               | 0.2       | SPAWAR               | First draft to begin discussion of architecture  |
| 09/21/2001               | 0.3       | TFWEB Team           | Second Draft to include architecture to date   |
| 10/05/2001               | 1.0       | TFW Team             | All sections updated to contain baseline information presented at Integrator's Conference  |
| 11/02/2001               | 2.0       | Portal Team          | 3.13.2 Added Service Guidelines for development of Templates<br>6.1.2 Added Portal Connector Stub Service Reverse Proxy Guidelines |
| 11/2/2001                | 2.0       | AMCS (John Stafford) | Complete revision of Application Owner guidance  |
|                          |           |                      |  |
|                          |           |                      |  |
|                          |           |                      |  |
|                          |           |                      |  |
|                          |           |                      |  |
|                          |           |                      |  |
|                          |           |                      |  |
|                          |           |                      |  |
|                          |           |                      |  |

## Revision Process

The contents of this document are of special interest to the TFWeb Integrators and those team members should be notified when any updates are made or related documents are created.

## Document Properties

Author(s): TFWeb – ISF Team

Publication Date: 11/09/2001

File Name: TFW\_Developer\_Guide\_v2.0.doc

This document was created using: MS Word 2000

## Table of Contents

|         |   |     |
|---------|---|-----|
| 1       | Introduction .....                                    | 1-1 |
| 1.1     | Intended Audience .....                               | 1-2 |
| 1.1.1   | Open Source Site .....                                | 1-2 |
| 1.2     | Purpose .....   | 1-2 |
| 1.3     | Assumptions .....                                     | 1-3 |
| 1.4     | Scope.....  | 1-4 |
| 1.5     | Document Structure .....                              | 1-5 |
| 2       | Levels of Integration.....                            | 2-1 |
| 2.1     | Hyperlink Integration .....                           | 2-1 |
| 2.2     | Presentation Integration.....                         | 2-1 |
| 2.3     | Application/Data Integration.....                     | 2-2 |
| 2.4     | Capability Comparison Across Integration Levels.....  | 2-2 |
| 2.5     | Integration Level Requirements Summary .....          | 2-3 |
| 3       | Enterprise Portal .....                               | 3-1 |
| 3.1     | Types Of Portals .....                                | 3-1 |
| 3.1.1   | TFWeb Policy .....                                    | 3-1 |
| 3.2     | TFWeb Portal Vision .....                             | 3-1 |
| 3.3     | Components of the Portal .....                        | 3-2 |
| 3.4     | Portal Engine .....                                   | 3-3 |
| 3.4.1   | Portal Runtime Environment.....                       | 3-3 |
| 3.4.1.1 | Servlet Engine .....                                  | 3-3 |
| 3.4.1.2 | Java Run Time Environment (JRE) .....                 | 3-3 |
| 3.4.1.3 | Java Plug-in.....                                     | 3-4 |
| 3.4.2   | Portal Server Features .....                          | 3-4 |
| 3.4.2.1 | Presentation and Rendering.....                       | 3-4 |
| 3.4.2.2 | Personalization and Customization .....               | 3-4 |
| 3.5     | Web Browsers.....                                     | 3-5 |
| 3.5.1   | Microsoft Internet Explorer .....                     | 3-5 |
| 3.5.2   | Netscape .....  | 3-5 |
| 3.5.3   | Personal Digital Assistants (PDAs) .....              | 3-5 |
| 3.5.4   | Cell Phones/Pagers.....                               | 3-6 |
| 3.5.5   | Handheld Scanners.....                                | 3-6 |
| 3.5.6   | Presentation Standards.....                           | 3-6 |
| 3.5.6.1 | Media Format Conventions .....                        | 3-6 |
| 3.6     | Portal Registry .....                                 | 3-7 |
| 3.7     | Portal Repository .....                               | 3-7 |
| 3.8     | How does Portal connect to the Application/Data ..... | 3-7 |
| 3.8.1   | Portal Connector .....                                | 3-7 |
| 3.8.2   | Portal Connectors vs. Services .....                  | 3-8 |
| 3.8.2.1 | Hyperlink Integration Requirements .....              | 3-8 |
| 3.8.2.2 | Presentation Integration Requirements .....           | 3-8 |
| 3.8.2.3 | Application/Data Integration Requirements .....       | 3-8 |
| 3.8.2.4 | Access Control .....                                  | 3-9 |

|            |  |      |
|------------|--|------|
| 3.9        | Portal Operational Flow .....  | 3-9  |
| 3.10       | Portal Development Kit (PDK) .....   | 3-11 |
| 3.10.1     | Portal Connector Template (PCT) .....                                      | 3-11 |
| 3.10.1.1   | Pseudo Code for the Portal Connector Template (PCT) .....                  | 3-12 |
| 3.10.2     | Portal Connector Stub (PCS) .....  | 3-12 |
| 3.10.3     | Application/Service Help.....  | 3-13 |
| 3.10.3.1   | Application Help Facilities .....  | 3-13 |
| 3.11       | Portal Look and Feel.....  | 3-13 |
| 3.11.1     | Description .....  | 3-14 |
| 3.11.2     | Portal Layout Definition .....   | 3-14 |
| 3.11.3     | Header Definition.....   | 3-14 |
| 3.11.3.1   | Branding for Command / User Group.....                                     | 3-14 |
| 3.11.3.2   | Portal Navigational Controls.....  | 3-14 |
| 3.11.3.2.1 | Home.....  | 3-15 |
| 3.11.3.2.2 | WorkPlaces .....   | 3-15 |
| 3.11.3.2.3 | InfoStore.....   | 3-15 |
| 3.11.3.2.4 | MyProfile .....  | 3-15 |
| 3.11.3.2.5 | Help .....   | 3-16 |
| 3.11.3.2.6 | Logoff .....   | 3-16 |
| 3.11.3.3   | Search Portal Library.....   | 3-16 |
| 3.11.3.4   | Header Example.....  | 3-16 |
| 3.11.3.5   | Header Development .....   | 3-17 |
| 3.11.3.6   | Portal Header Templates .....  | 3-18 |
| 3.11.3.7   | Procedure to Create New Portal Templates.....                              | 3-18 |
| 3.11.3.8   | Content Definition.....  | 3-20 |
| 3.11.3.9   | Home Page .....  | 3-20 |
| 3.11.4     | Footer Definition .....  | 3-20 |
| 3.11.5     | Portal Example .....   | 3-21 |
| 3.12       | Standards .....  | 3-21 |
| 3.13       | Administration and Management .....  | 3-22 |
| 3.13.1     | Portal Administration .....  | 3-22 |
| 3.13.2     | Theme (Template) Administration .....                                      | 3-23 |
| 3.13.3     | Portal Connector Development .....   | 3-24 |
| 3.14       | Roles / Responsibilities of a Portal Administrator.....                    | 3-24 |
| 3.14.1     | Workgroup (Role) Administration .....                                      | 3-24 |
| 3.14.2     | User Administration .....  | 3-24 |
| 3.14.3     | Template Administration.....   | 3-24 |
| 3.14.4     | Knowledge (content) administration / publishing. ....                      | 3-24 |
| 3.14.5     | Channel Administration. ....   | 3-24 |
| 3.14.6     | NMCI services administration.....  | 3-24 |
| 3.15       | Roles / Responsibilities of a Portal Developer .....                       | 3-24 |
| 3.15.1     | Develop Themes .....   | 3-24 |
| 3.15.2     | Incorporate Portal Templates with Services .....                           | 3-25 |
| 3.15.2.1   | Include the URL path used to reference the Cascading Style<br>Sheets ..... | 3-25 |

|            |  |      |
|------------|--|------|
| 3.15.2.2   | Include tags (selectors/elements) in applications to define the attributes to be implemented ..... | 3-25 |
| 3.15.3     | Portal Friendly Service Development .....  | 3-30 |
| 3.15.3.1   | Images .....   | 3-30 |
| 3.15.3.2   | Links.....   | 3-31 |
| 3.15.3.2.1 | Links for Integration Level 1: .....   | 3-31 |
| 3.15.3.2.2 | Links for Integration Level 2 & 3: .....   | 3-31 |
| 3.15.3.2.3 | Form Action .....  | 3-31 |
| 3.16       | Portal Connector and Service Integration Process .....   | 3-32 |
| 3.17       | How will User Browse / Access the Portal Connector .....   | 3-33 |
| 4          | Web Service-Based Architecture .....   | 4-1  |
| 4.1        | Key Concepts .....   | 4-1  |
| 4.2        | Application and Information Services.....  | 4-1  |
| 4.3        | Service Definition .....   | 4-2  |
| 4.4        | Service Repository .....   | 4-2  |
| 4.4.1      | Enterprise-Level Service Repository .....  | 4-3  |
| 4.4.2      | Claimant and Local -Level Service Repositories .....   | 4-3  |
| 4.5        | Service Registry.....  | 4-3  |
| 4.6        | Portal to Repository Interface .....   | 4-6  |
| 4.6.1      | Transport Protocol.....  | 4-6  |
| 4.6.1.1    | Support for Standard HTTP/HTTPS headers .....  | 4-6  |
| 4.6.2      | HTTPS Request .....  | 4-6  |
| 4.6.2.1    | Request Data Definition .....  | 4-7  |
| 4.6.2.2    | Timeouts .....   | 4-8  |
| 4.6.2.2.1  | HTTPS Response .....   | 4-8  |
| 4.6.2.3    | Response Data Definition.....  | 4-8  |
| 4.6.2.4    | Content Requirements .....   | 4-9  |
| 4.6.2.4.1  | Error Handling .....   | 4-9  |
| 5          | Enterprise Portal Taxonomy.....  | 5-1  |
| 6          | Security .....   | 6-1  |
| 6.1        | Security Definitions .....   | 6-1  |
| 6.2        | TFWeb Security Architecture .....  | 6-1  |
| 6.2.1      | Role of the Directory in SSO .....   | 6-2  |
| 6.2.2      | SSO Architecture Components .....  | 6-3  |
| 6.2.3      | How TFWeb SSO Works.....   | 6-4  |
| 6.3        | TFWeb Security Model - Allow then Deny .....   | 6-4  |
| 6.4        | Application Access Control Mechanisms .....  | 6-5  |
| 6.4.1      | ClearTrust Application Access Control Mechanisms .....   | 6-5  |
| 6.4.1.1    | User Passwords .....   | 6-5  |
| 6.4.2      | Legacy Application Access Control Mechanisms .....   | 6-6  |
| 6.4.3      | Legacy Application Authentication Options .....  | 6-6  |
| 6.5        | Application Interactions with SSO/AD .....   | 6-7  |
| 6.5.1      | Authorizing User's Access to the Application .....   | 6-7  |
| 6.5.2      | HTTP Headers .....   | 6-8  |
| 6.5.3      | ClearTrust Web Server Plug-Ins.....  | 6-9  |
| 6.6        | Limits to TFWeb Single Sign-On.....  | 6-9  |

|         |   |     |
|---------|---|-----|
| 7       | Application/Service Integration.....                              | 7-1 |
| 7.1     | Assumptions and Limitations .....                                 | 7-1 |
| 7.1.1   | Standards Based Approach.....                                     | 7-1 |
| 7.1.2   | Bandwidth Limitations .....                                       | 7-1 |
| 7.1.3   | Security .....  | 7-2 |
| 7.1.4   | Service Module Definition.....                                    | 7-2 |
| 7.1.5   | Local/Claimant vs. Enterprise Service Repository.....             | 7-2 |
| 7.2     | Application Packaging Requirements .....                          | 7-2 |
| 7.2.1   | Application/Data Adaptor (Component Software).....                | 7-2 |
| 7.2.2   | Service Registry Metadata Information .....                       | 7-3 |
| 7.2.3   | XML Schema.....   | 7-3 |
| 7.3     | Integration Examples .....  | 7-3 |
| 7.3.1   | Hyperlink Integration .....                                       | 7-3 |
| 7.3.2   | Presentation Integration .....                                    | 7-3 |
| 7.3.3   | Application/Data Integration .....                                | 7-4 |
| 7.3.3.1 | Parse PRI Request.....  | 7-4 |
| 7.3.3.2 | Make Application Request.....                                     | 7-4 |
| 7.3.3.3 | Application Accepts Request.....                                  | 7-4 |
| 7.3.3.4 | PRI Response .....  | 7-4 |
| 7.3.3.5 | Application Results Return .....                                  | 7-4 |
| 7.4     | Messaging Protocols.....  | 7-4 |
| 7.5     | Management.....   | 7-5 |
| 7.5.1   | Service Registry .....  | 7-5 |
| 7.5.2   | Component Repository.....   | 7-5 |
| 7.5.3   | Session Management.....   | 7-6 |
| 7.5.4   | Replication.....  | 7-6 |
| 8       | Service Certification Process .....                               | 8-1 |
| 8.1     | Process Overview .....  | 8-1 |
| 8.2     | Information Assurance Certification and Accreditation.....        | 8-2 |
| 8.3     | Service Intent to Migrate, Rationalization, and Registration..... | 8-2 |
| 8.3.1   | Registry Metadata .....   | 8-2 |
| 8.3.2   | Service Repository Package .....                                  | 8-3 |
| 8.4     | TFWeb Beta Lab Test.....  | 8-3 |
| 8.4.1   | Process .....   | 8-5 |
| 8.5     | NMCI Application Certification Process .....                      | 8-6 |
| 8.5.1   | Process .....   | 8-6 |
| 8.6     | WEN IT Governance.....  | 8-7 |
| 9       | Coding Standards, Policies and Guidelines .....                   | 9-1 |
| 9.1     | Directory Structure and Variable Naming Conventions.....          | 9-1 |
| 9.1.1   | Assumptions.....  | 9-1 |
| 9.1.2   | Directory Structure .....   | 9-1 |
| 9.1.3   | Filename Standards .....  | 9-2 |
| 9.2     | Variable Management.....  | 9-2 |
| 9.2.1   | General Naming Conventions for Variables .....                    | 9-2 |
| 9.2.2   | Local Variables.....  | 9-2 |
| 9.2.3   | Global Variables .....  | 9-3 |

|            |  |      |
|------------|--|------|
| 9.2.4      | Permanent Client Side Cookies.....                         | 9-3  |
| 9.2.5      | Server Side Session Variables.....                         | 9-4  |
| 9.3        | Standard Error Trapping.....                               | 9-4  |
| 9.4        | JSP Standards.....   | 9-4  |
| 9.4.1      | JSP Error/Exception Handling.....                          | 9-4  |
| 9.4.2      | Environment Cleanup.....                                   | 9-5  |
| 9.5        | CGI Standards.....   | 9-5  |
| 9.5.1      | CGI Error/Exception Handling.....                          | 9-5  |
| 9.5.1.1    | C.....   | 9-5  |
| 9.5.1.2    | Perl.....  | 9-5  |
| 9.5.2      | Environment Cleanup.....                                   | 9-5  |
| 9.6        | ASP Standards.....   | 9-6  |
| 9.6.1      | Network Bandwidth Issues.....                              | 9-6  |
| 9.6.2      | Error/Exception Handling.....                              | 9-6  |
| 9.6.3      | Environment Cleanup.....                                   | 9-6  |
| 10         | Development Tools and Resources.....                       | 10-1 |
| 10.1       | Development Tools.....                                     | 10-1 |
| 10.2       | Testing Tools.....   | 10-1 |
| 10.2.1     | Portal Connector Stub (PCS).....                           | 10-1 |
| 10.2.1.1   | PCS Layout.....  | 10-2 |
| 10.2.1.2   | PCS Components.....  | 10-4 |
| 10.2.1.2.1 | System Files (Non-Modifiable).....                         | 10-4 |
| 10.2.1.3   | Data Files.....  | 10-5 |
| 10.2.1.3.1 | Input File(Modifiable).....                                | 10-5 |
| 10.2.1.3.2 | Output File.....   | 10-5 |
| 10.2.1.4   | How To Test a Service.....                                 | 10-5 |
| 10.2.1.5   | How to Install PCS.....                                    | 10-5 |
| 11         | Application Owner/Analyst Guidance.....                    | 11-1 |
| 11.1       | Pre-Service Registration Phase.....                        | 11-1 |
| 11.1.1     | Determining TFWeb Integration Goals.....                   | 11-1 |
| 11.1.1.1   | Determining Communities of Interest.....                   | 11-2 |
| 11.1.2     | Reviewing Existing Services.....                           | 11-2 |
| 11.1.2.1   | Market Review of Existing Services and Content.....        | 11-2 |
| 11.1.2.2   | Registered Services and "Best of Breed" Determination..... | 11-2 |
| 11.1.3     | Supportability and Maintainability.....                    | 11-3 |
| 11.1.4     | Web Enablement Determination.....                          | 11-3 |
| 11.1.4.1   | Existing Web-Enabled Applications.....                     | 11-3 |
| 11.1.4.2   | Non-Web-Enabled Applications.....                          | 11-3 |
| 11.1.4.2.1 | Information Services.....                                  | 11-4 |
| 11.1.4.2.2 | Real-Time Versus Non-Real-Time.....                        | 11-4 |
| 11.1.4.2.3 | Service/Application User Environment.....                  | 11-4 |
| 11.1.4.2.4 | User/Administrator.....                                    | 11-4 |
| 11.2       | Intent to Migrate.....                                     | 11-4 |
| 11.2.1     | Submission to the application information database.....    | 11-5 |
| 11.2.2     | Integration Level Appropriateness.....                     | 11-5 |

|   |      |
|---|------|
| 11.2.3 Identify if the program uses Java, JavaScript, ActiveX, or plugins .....   | 11-5 |
| 11.2.4 Examine the application database for similar programs that are currently under development .....   | 11-5 |
| 11.2.5 Determine current security model and whether IATO/ATO exists, or is required .....   | 11-5 |
| 11.2.6 Determine XML integration requirements .....   | 11-5 |
| 11.3 Service Registration .....   | 11-5 |
| 11.3.1 Verify completeness and accuracy of portal metadata .....  | 11-6 |
| 11.3.2 Verify migration plan for level of integration is submitted .....  | 11-6 |
| 11.3.3 Ensure IATO/ATO has been updated if security model changed for TFW migration .....   | 11-6 |
| 11.3.4 Verify initial access control list submitted along with information describing method of updating ACL .....  | 11-6 |
| 11.3.5 Portal Compliance Testing .....  | 11-6 |
| 11.3.6 Review summary of testing accomplished .....   | 11-6 |
| 11.3.7 Review portal integration information submitted .....  | 11-6 |
| 11.3.8 DoN XML guideline compliance .....   | 11-6 |
| 11.3.9 Set next review date .....   | 11-7 |
| 11.3.10 Verify database entry is complete and accurate in AMCS application database .....   | 11-7 |
| 11.3.11 Technical Review .....  | 11-7 |
| 11.3.12 Configuration Verification .....  | 11-7 |
| 11.3.13 Ensure application is logged in the DON CIO Data Management and Interoperability Repository .....   | 11-7 |
| 11.3.14 Verify all application data structures and data interfaces are documented .....   | 11-7 |
| 11.3.15 Verify AMCS OIC has approved migration plan for application/data overlap .....  | 11-7 |
| 11.3.16 Ensure developer requirements for future capability upgrades of theWEN architecture are documented if current implementation does not allow for desired developer functionality ..... | 11-7 |
| 11.4 Application/Service Delivery Phase .....   | 11-8 |
| 11.4.1 Application Acceptance .....   | 11-8 |
| 11.4.2 Application Delivery .....   | 11-8 |
| 11.4.3 Application Integration .....  | 11-8 |
| 11.4.3.1 Developer Integration Environment .....  | 11-8 |
| Appendix A: Department of Defense and Department of the Navy Authority References .....   | 1    |
| Appendix B: Service Code Samples/Templates .....  | 1    |
| Appendix C: Standards .....   | 1    |
| Appendix D: Development References .....  | 1    |
| Appendix E: Terminology Glossary .....  | 1    |



## Table of Figures and Tables

|  |      |
|--|------|
| Figure 1-1: Three Tiers of the TFWeb Portal.....   | 1-1  |
| Table 2-1. Capabilities By Levels of Integration .....                                     | 2-3  |
| Table 2-2. Integration Level Requirements .....  | 2-3  |
| Figure 3-1: TFWeb Vision .....   | 3-2  |
| Figure 3-2: Portal Framework Components .....  | 3-3  |
| Figure 3-3: Presentation and Rendering .....   | 3-4  |
| Figure 3-4: Portal Operational Flow .....  | 3-10 |
| Figure 3-5: Portal Header Design Template.....   | 3-16 |
| Figure 3-6: Portal Header Example.....   | 3-17 |
| Figure 3-7: Template Examples .....  | 3-18 |
| Figure 3-8: Footer Example.....  | 3-20 |
| Figure 3-9: Portal Example.....  | 3-21 |
| Figure 3-10: The Jasmine portal Administration.....  | 3-22 |
| Figure 3-11: The Jasmine template Administration .....                                     | 3-24 |
| Table 3-12: Style Tag Descriptions for Style Sheets .....                                  | 3-26 |
| Figure 3-13: Style Sheet diagram for Infostore.....  | 3-29 |
| Figure 3-14: Style Sheet diagram for Workplaces.....                                       | 3-30 |
| Figure 3-15: Portal Connector Library Screen Shot .....                                    | 3-34 |
| Figure 4-1: TFWeb Repository Architecture.....   | 4-1  |
| Figure 4-2: Enterprise Service Repository.....   | 4-3  |
| Figure 4-3: Enterprise Service Registry .....  | 4-4  |
| Figure 4-4: Relationship between Registry Metadata and Repository Service<br>Modules ..... | 4-5  |
| Figure 4-5: Application/Data Integration Process Flow Diagram .....                        | 4-5  |
| Figure 4-6: Portal to Repository Interface .....   | 4-6  |
| Table 4-7: PRI Request Data Definition .....   | 4-7  |
| Table 4-8: PRI Response Data Definition.....   | 4-8  |
| Figure 5-1: Service-Centric Access.....  | 5-1  |
| Table 5-2: Initial TFWeb Portal Taxonomy .....   | 5-1  |
| Figure 6-1: TFWeb Security Architecture .....  | 6-2  |
| Figure 7-1 Standards Based Architecture .....  | 7-1  |
| Figure 7-2 Legacy Data Presentation Examples .....   | 7-4  |
| Figure 8-1 Service Certification Process. ....   | 8-1  |
| Figure 8-2 TFWeb Beta Test Process.....  | 8-3  |
| Figure 8-3 NMCI Certification Lab Process.....   | 8-6  |
| Figure 10-1: Content Frame Layout (100%).....  | 10-2 |
| Figure 10-2: Content Frame Layout (50:50).....   | 10-3 |
| Figure 10-3: Content Frame Layout (30:70).....   | 10-3 |
| Figure 10-4: PCS.....  | 10-6 |
| Figure 10-5: Extracted Data from the Cookie Fields.....                                    | 10-7 |

## Executive Summary

### *TFW Mission*

*"To provide integrated and transformational information exchange for both the ashore and afloat navy to take full advantage of Navy's IT21 and NMCI infrastructure investments."*

Task Force Web (TFWeb) has been stood up to implement the vision for a Web Enabled Navy. That vision is realized by an enterprise three-tiered architecture: presentation, application, and data. This guide focuses on the data tier and its integration with the application tier. It is assumed that all analysis and requirements gathering is complete prior to initiating this development effort.

This document provides key information for Navy service owners (i.e., developers, integrators, and implementers) of operational and business processes to enable integration of existing service applications into the Enterprise Portal infrastructure. The goal of this document is to provide sufficient detail so that integration problems are minimized. Wherever this document does not provide specific direction, references are provided to appropriate guidance.

This guide documents a rapidly evolving environment. As such, it is a work in progress and will change over time as the technology is enhanced and as additional integration issues are identified. Each section will be expanded as more information becomes available and guidance to developers/implementers is generated.

# 1 Introduction

On 28 August 2000, Vice Chief of Naval Operation (VCNO), issued a memorandum subject: *Software/Applications for NMCI*. A key element identified in this memo was “a simple but clear definition of what a web enabled Navy would be.” Numerous organizations participated in refining a common technical understanding of high-level issues involved in web enabling the Navy. Throughout October and November 2000, key requirements and broad timelines were explored. In December 2000, VCNO chartered Task Force Whiskey (TFW) to provide within 60 days a detailed analysis and a workable execution strategy to web enable the Navy. The final report provided 31 January 2001 provided a vision for operational, technical, and system architectures as well as a possible timeline.

Task Force Web (TFWeb) has been established to implement a web-enabled Navy. That vision is enabled by an enterprise three-tiered architecture describing where the different web technologies reside. The three tiers are presentation, application, and data (see Figure 1-1). The communication between the layers is based on standard APIs and open-industry standards, such as XML. It leverages various standard NMCI and TFW Afloat horizontal services. The proposed architecture is a comprehensive solution that provides a number of benefits, including:

- ÷! Multiple physically distributed portals but one logical portal.
- ÷! Load balancing and clustering will provide scalability and high availability.
- ÷! An open, flexible, distributed, and extensible architecture.
- ÷! Minimized risk of introducing new technologies.
- ÷! A strategic foundation for business and process integration to provide a consolidated view of the enterprise.

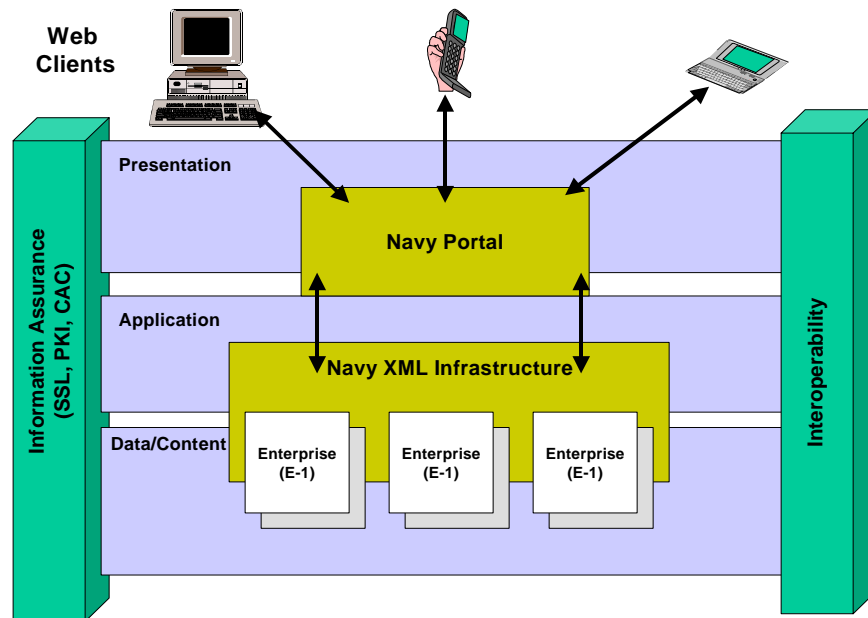


Figure 1-1: Three Tiers of the TFWeb Portal

## 1.1 Intended Audience

This guide is focused on the data tier and its integration with the application tier. Other documents will address other aspects of web enablement. Thus, this guide is intended for the following audience:

- ÷! WEN Integration Architects/Engineers
- ÷! WEN Application Developers
- ÷! TFWeb Integration Architects/Engineers

### 1.1.1 Open Source Site

In addition to this document, a valuable source of information is the TFW Open Source Site (OpSS). This website, itself a portal, is a place to find information, ask questions, and see what other developers have contributed. The URL below provides minimal access as a guest. Register for full access.

<https://tfw-opensource.spawar.navy.mil>

## 1.2 Purpose

The purpose of this document is to capture sufficient technical information for Navy service owners (i.e., developers, integrators, and implementers) of both operational and business processes to enable integration of existing service applications into the Enterprise Portal infrastructure. The goal of this document is to provide sufficient detail so that integration problems can be minimized. Wherever this document does not provide specific guidance, specific references will be provided to where the appropriate guidance can be found.

It is assumed that all analysis and requirements gathering is complete prior to initiating this development effort, therefore this guide will only focus on those technical elements required for integration of services into the portal. This document does not address the decision of when to create a service, or the selection of specific application functionality to incorporate within the service. Figure 1-2 identifies the areas of the portal architecture that are addressed in this guidance and that impact the service provider.

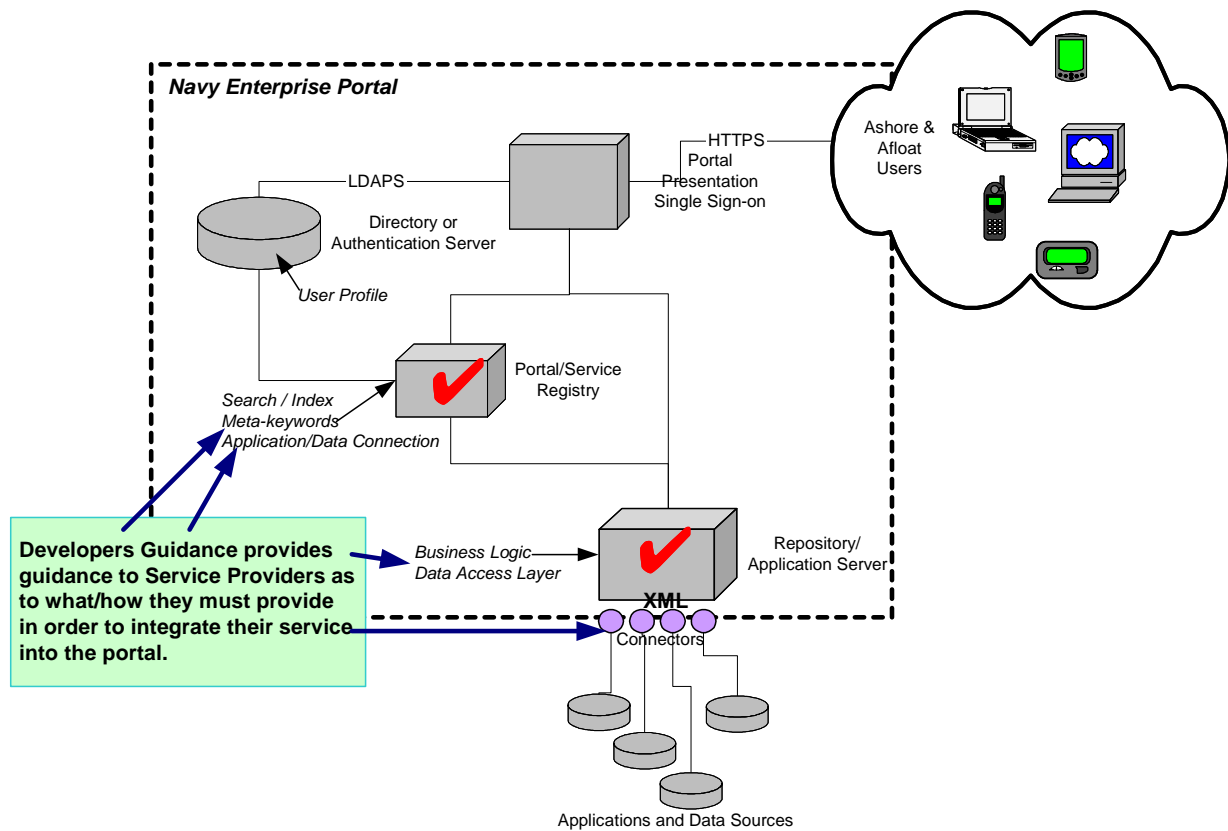


Figure 1-2: Overall TFWeb Enterprise System Architecture

It is assumed that the applications being considered for integration with the Enterprise portal have been web enabled. While this document is focused on the integration to the enterprise portal and does not describe in detail how to web enable the application, it does provide high-level guidance a developer must consider before migrating their applications. As described in Section 2, there are increasing levels of integration, and simple web enabling of an application (i.e., access to an application through a web-browser client) only achieves hotlink integration with the enterprise portal, which is the least desirable level of integration.

### 1.3 Assumptions

The following assumptions are made regarding the environment in which this Enterprise portal will be implemented.

- ÷! **Scope** – The capabilities of the Enterprise portal will be deployed to ashore and afloat organizations and facilities. The Enterprise Portal will leverage NMCI and TFW Afloat infrastructures. The enterprise portal provides access to the content and data, which are accessible through web-enabled applications.
- ÷! **Focus** – The focus of the Enterprise portal is initially internal. That is, initial functionality will be developed and deployed to support Navy/Marine Corps business and warfare operations processes. The middle and long-term possibilities include: use of the portal system to inter-operate with Allies, coalition forces, commercial suppliers, retirees, and dependents. The Enterprise Portal will be structured to host or link to other service (Joint/non-DoD) applications whenever possible.

- ÷! **Implementation** – Implementation and deployment of web-service capabilities will be incremental; that is, delivery of functionality and content to the users will be evolutionary. This concept of operations is expressed from the viewpoint of a steady state, fully functional enterprise portal system, where the information content of the system is ever changing.
- ÷! **Commercial Standards-Centric** – The Enterprise Portal will use commercial standards for as much of its interfaces, underlying technologies, and applications as possible. Primary examples of technologies or interfaces to be used can be described (though not limited to) as the following:
  - . ! **Java/Java 2 Enterprise Edition (J2EE)**: Java was introduced in 1995 by Sun Microsystems. It is an object-oriented language designed for the World Wide Web, similar to C/C++, in which the source is compiled into 'bytecode', which is then interpreted by run-time environment (known as a Java Virtual Machine) on the host machine. J2EE is a java-centric environment for developing and deploying multi-tiered web-based applications. Some key features of J2EE include:
    - **Java DataBase Connectivity (JDBC)**. JDBC the Java equivalent to ODBC, which acts as the standard database connection language/method for Java applications.
    - **Enterprise JavaBeans (EJB)**. EJB is a Java API developed by Sun that defines the component architecture for multi-tiered systems. EJBs are the objects in a multi-tiered object-oriented J2EE environment, and enable the developers to focus on actual business architecture as opposed to developing the interfaces between the different components themselves.
    - **Servlets and Java Servlet Pages (JSPs)**. Servlets are java applets that run on the server side as an alternative to Common Gateway Interface (CGI) applications. JSPs are an extension of servlets that allow web developers to dynamically build web pages.
  - . ! **eXtensible Markup Language (XML)**: XML is an extension/subset of Standard Graphical Markup Language (SGML) specifically designed for WWW dissemination and display of data. It is an open framework in which developers can develop (and, more importantly, standardize and validate against) a tagged data format. When done properly, the tagging becomes a form of metadata that can be used to more easily transition/translate the data inter/intra system or application. Furthermore, the rendering of XML is detached from the data itself. Therefore, the data in an XML document/file can be reformatted for display or processing any number of ways without ever having to modify the tags they contain.
- ÷! **NMCI** - The Navy & Marine Corps Intranet (NMCI) will supply the shore-based and embarkable infrastructure.
- ÷! **TFW Afloat** – TFW Afloat and related shipboard standards and delivery mechanisms will provide an afloat infrastructure.

## 1.4 Scope

This document is intended to be read by individuals with service or application development and maintenance responsibilities who are tasked to integrate their web-enabled service or application into the Enterprise portal. The scope of this document is limited to the technical integration issues with accomplishing this goal. Other policy and guidance information is referenced in the DoD Authority References (see Appendix A).

This guide documents a rapidly evolving environment. As such, it is a work in progress and will change over time as the technology is enhanced and as additional integration issues are identified. Each section will be expanded as more information becomes available and guidance

to developers/implementers is generated. The latest copy of this guide, tools referenced within it and other pertinent documentation may be found on the TFWeb OpSS.

This guidance will answer the question, "How do I get my application into the operational Enterprise portal environment?" by providing how to perform the following:

- ÷! Register a developer for access to the TFWeb environment.
- ÷! Register and query applications, data structures/fill, Object Interfaces/Application Programming Interfaces (APIs).
- ÷! Schedule integration and operational testing.

Additionally, the role of the TFWeb team with regards to the portal integration development process will be as follows:

- ÷! Establish registry requirements for applications served by the Enterprise portal (e.g., applications, content).
- ÷! Establish and track metrics to ensure:
  - . ! Timeliness and availability of updates for specific applications
  - . ! Consolidation of applications
  - . ! Bandwidth and storage projection
- ÷! Generate and maintain the technology implementation/migration plan.
- ÷! Establish security policy.
- ÷! Act as Designated Approving Authority (DAA) for the Enterprise portal testing, beta and pilot environments, as necessary.

## 1.5 Document Structure

This document has been structured in a logical manner to address the questions that will be asked by service providers with greater and greater levels of detail. The sections of the document are as follows:

- ÷! Section 2 defines a set of levels of web and portal integration and highlights the goal state for portal integration.
- ÷! Section 3 provides in-depth details about each element of the enterprise portal: Portal Engine, Web Browsers, Registry, Repository, Portal Connectors, Portal Development Kit.. It also provides guidance on how to connect to the portal, tools to assist with this and standards to follow.
- ÷! Section 4 describes the process for first identifying a service to be migrated, as well as the associated processes to get an application or service integrated into the Enterprise portal.
- ÷! Section 5 describes the taxonomy for the Enterprise portal so that service providers and content managers can determine where to integrate their information sources into the Enterprise portal
- ÷! Section 6 provides information on the portal security architecture including Single Sign On (SSO).
- ÷! Section 7 describes in further detail the levels of integration mentioned in Section 2. It provides processes and procedures that allow the testing and certification of applications within the portal environment.
- ÷! Section 8 provides the process for certifying services within the portal environment.

- ÷! Section 9 introduces industry standard policies and guidelines for code to assist developers in choosing best of breed solutions.
- ÷! Section 10 identifies development tools of interest to service providers in web enabling and integrating with the Enterprise portal. This document will not specify or necessarily recommend specific tools, but will certainly identify specific tools that through experience have been shown to be inadequate.
- ÷! Section 11 provides the process for migrating an existing application into the portal, ensuring that it meets all standards and does not duplicate existing services.
- ÷! Appendix A identifies all the related Department of Defense (DoD) and Department of the Navy (DON) Authority References and other TFWeb-related documents.
- ÷! Appendix B provides templates and sample code for JSP, ASP, XML, XSL and CGI.
- ÷! Appendix C addresses standards enforced by the TFWeb Enterprise Portal solution and the DoN.
- ÷! Appendix D lists references that provide further reading on tools and standards used in the FTWeb Enterprise Portal solution.
- ÷! Appendix E provides a list of terms and acronyms used in this document or other documents referenced within it.



## 2 Levels of Integration

Levels of integration are defined to be the degree of integration between a service or application and the enterprise portal. Web enabling of an application will provide some degree of access to an application through a web browser, but the levels of integration defined here describe the degree to which the web-enabled application has been integrated into the portal.

Application/Data Integration is the desired level of integration; however, there may be justifiable reasons why that level of integration cannot be achieved or does not make sense for a particular application. Rationale will need to be provided for applications that will only achieve Hyperlink or Presentation levels of integration.

Each level of integration defined below will have specific requirements for integrating with the enterprise portal. While this section defines the levels of integration and the integration goal, Section 3.2.1 will describe the integration requirements and process for each of these levels of integration. Please note that this discussion refers to the integration of services/applications with the TFWeb Portal.

### 2.1 Hyperlink Integration

Hyperlink Integration provides “as is” access to an existing web-based application through a hyperlink or a list of hyperlinks displayed within a pane of the portal. The hyperlink is created as a service within the service repository, with access to this service controlled through the permissions management application. When accessed, the hyperlink is displayed within a pane of the portal on the user’s desktop, and generally includes a brief description of the application being accessed, and possibly an associated lightweight graphic. When the user selects the hyperlink, a new browser window is opened on the user’s desktop, through which the application will execute. At this point, communication occurs directly between the new browser window and the application.

This method is usually easiest to implement, though it is the least desirable. Application owners attempting this level of integration must first obtain a waiver from TFWeb, and must also provide a migration plan for achieving a higher level of integration. Hyperlink Integration requires no true integration with enterprise portal services or content, other than the initial authentication that the user has access to the service containing the hyperlink. Therefore, there is minimal benefit gained in its integration into the enterprise portal. When working with multiple applications using hyperlink integration, multiple browsers will be opened on the user’s desktop.

Hyperlink Integration is commonly referred to as Level 1 Integration.

### 2.2 Presentation Integration

Presentation Integration provides “as is” access to an already web-enabled application. This level of integration requires that all application content be rendered within a pane of the portal. The initial connection to the application is created as a service module in the Enterprise Service Repository utilizing a standard HTTP redirect to the application. Access to the service module is controlled through the permissions management application. When accessed by the user, the application is displayed within a pane of the portal on the user’s desktop. The user is able to directly interact with the application appearing in this pane. All communication between the user and the application must flow through the portal. This type of communication is implemented through a reverse HTTP proxy mechanism within the enterprise portal, and requires that the application present its content in portal compliant HTML or XML/XSL. The definition of portal-compliant HTML or XML/XSL is defined in a later section of this document.

Application owners attempting this level of integration do not need to obtain a waiver from TFWeb, but they must provide a migration plan for achieving a higher level of integration.

Presentation Integration allows multiple applications to be visible within multiple panes (e.g., channels, frames) of the same browser window of the enterprise portal. However, the content and data access mechanisms still reside outside of the service repository, and therefore may not match the presentation rendering guidelines of TFWeb.

Presentation Integration is commonly referred to as Level 2 Integration.

## 2.3 Application/Data Integration

Application/Data Integration involves a more closely coupled integration of the application with the enterprise portal. Application/Data Integration is the TFWeb-preferred level of integration. All application content is provided through services that reside in the service repository. These types of services act as lightweight connectors, exposing some portion of application functionality in a manner that is compliant with the enterprise portal. Application logic continues to reside within the application/data layers, and not within the service. When accessed by a user, all application content is rendered within a pane of the portal on the user's desktop. Access to all services is controlled by the permissions management application. The user is able to directly interact with the application appearing in this pane. All communication between the user and the application must flow through both the portal and the service repository.

When invoked, a service decomposes the request, accesses the appropriate application or data source to process the request, formats the results of the request into the appropriate portal-compliant HTML or XML/XSL response, and returns the response to the enterprise portal. The definition of portal-compliant HTML or XML/XSL is provided in a later section of this document. A description of the service will be registered with the global service registry to provide the enterprise portal with quick access and search capability. Section 8 provides service developers with additional details of how to build Application/Data Integration services.

Application/Data Integration is commonly referred to as Level 3 Integration.

## 2.4 Capability Comparison Across Integration Levels

There are a number of capabilities that are desirable for the enterprise portal, and the level of integration achieved for an application or service may impact the degree to which a particular capability can be achieved. A finite set of desirable capabilities are defined as follows:

- ÷! Single Sign-on – The ability for users to log on once to the portal and be able to access all authorized resources within the enterprise. A single point sign-on accepts the user's name and password and automatically logs on to all appropriate services after first capturing the user's individual names and passwords for each authorized resource.
- ÷! Access to Service – The ability to obtain access to a service from the portal.
- ÷! Service Presented in Portal – The ability to view/manipulate some portion of the service's information from within the portal's presentation with the assumption that multiple services can be viewed in the same portal presentation at the same time.
- ÷! Policy-enforced Look-and-Feel – The manual enforcement of standards by publishing policies and manually checking for compliance.
- ÷! Automatically enforced Look-and-Feel – The enforcement of standards by being the single point of control. In the case of the portal, if the portal is uniquely responsible for all presentation of information, then the look-and-feel can be automatically enforced by ensuring that the portal's presentation meets the standard.
- ÷! Data Sharing (cut-n-paste) – Data or information from one service is simply captured and then pasted into another service with the user taking responsibility for the format and definition of the data (e.g., cutting and pasting a text string). There is no understanding of the data by the portal. The portal processes information presentation (e.g., HTML) without having to understand the underlying data definition.

- ÷! Data Sharing (data aggregation) – The portal processes the information and is responsible for formatting the presentation (e.g., XML style sheets) and provides the potential for additional business logic that manipulates data from multiple sources and data aggregation.
- ÷! Business Process Integration – The re-engineering of existing business processes through the aggregation of services and data.

Table 2-1 shows the degree to which these capabilities can be achieved across the levels of integration. Clearly, application/data integration is the goal, but significant capability can be achieved with presentation integration. While hyperlink integration is not desirable, it may be an acceptable first step from web enabling to portal integration.

| Capabilities                         | Hyperlink | Presentation | Application |
|--------------------------------------|-----------|--------------|-------------|
| Single Point Sign-on                 | ✓         | ✓            | ✓           |
| Access to Service                    | ✓         | ✓            | ✓           |
| Service Presented in Portal          |           | ✓            | ✓           |
| Policy-enforced Look-and-Feel        | ✓         | ✓            | ✓           |
| Automatically-enforced Look-and-Feel |           | Limited      | ✓           |
| Data Sharing (cut-and-paste)         | Limited   | ✓            | ✓           |
| Data Sharing (data aggregation)      |           |              | ✓           |
| Business Process Integration         |           |              | ✓           |

**Table 2-1. Capabilities By Levels of Integration**

## 2.5 Integration Level Requirements Summary

Some of the most significant requirements for each level of integration are summarized in Table 2-2.

| Integration Level               | Service Module Requirement                            | HTTP Proxy | Application Output | Waiver Required | Migration Plan Required |
|---------------------------------|---|------------|--------------------|-----------------|-------------------------|
| 1. Hyperlink                    | HTML or XML/XSL                                       | No         | HTML or XML/XSL    | Yes             | Yes                     |
| 2. Presentation                 | ASP, JSP, or CGI containing HTTP Redirect             | Yes        | HTML or XML/XSL    | No              | Yes                     |
| 3. Application/Data Integration | ASP, JSP, or CGI with PRI and SOAP interface handlers | Yes        | SOAP/XML           | No              | No                      |

**Table 2-2. Integration Level Requirements**

## 3 Enterprise Portal

In general, an enterprise portal is a web-based desktop that provides a single, secure interface into the applications, information and processes of the enterprise not only for its employees but also for its customers, partners and clients for today's Internet business generation. It truly provides convergence of Web information and desktop applications through application integration.

### 3.1 Types Of Portals

- ÷! Vertical Portal: Service-centric portal built around a specific service offering or application or business function like ERP, SCM, CRM applications(e.g., functionality is built around the functionality of the particular application such as SAP).
- ÷! Horizontal Portal: Provides general information, no particular focus, variety of subjects, variety of users and cuts across business functions and applications. Examples: Lycos, Infoseek, Yahoo. Enterprise Portals: Enable an enterprise to share information and provide web-based access to the applications/services to its employees, customers and partners.
- ÷! Specialized Portals – Portals specific to a type of business e.g.
  - Supply Chain Portal: Specific to supply chain management business
  - E-tailing Portal: Specific to e-tailing business

#### 3.1.1 TFWeb Policy

As per the TFWeb policy there will be only one Enterprise Portal for the Department Of Navy. There will NOT be any Vertical, Horizontal or Specialized portals. All applications must interface with the single Enterprise Portal.

### 3.2 TFWeb Portal Vision

#### TFWeb Mission

*"To provide integrated and transformational information exchange for both the ashore and afloat navy to take full advantage of Navy's TFW Afloat and NMCI infrastructure investments."*

The communication between the layers is based on standard APIs and open-industry standards, such as XML. The communication leverages various standard NMCI and TFW Afloat horizontal services. The basic components and services of the services-based architecture are shown in following figure.

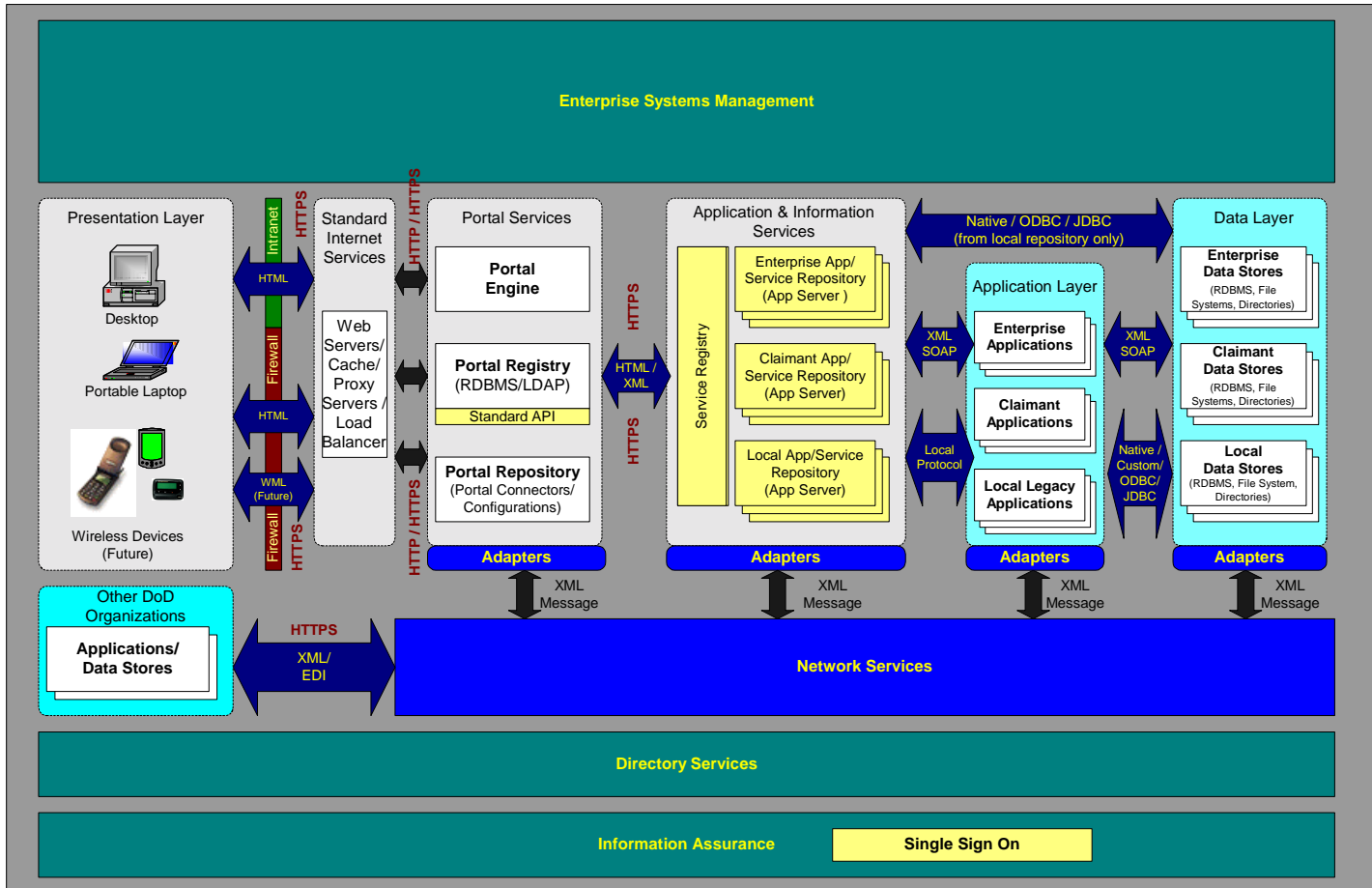


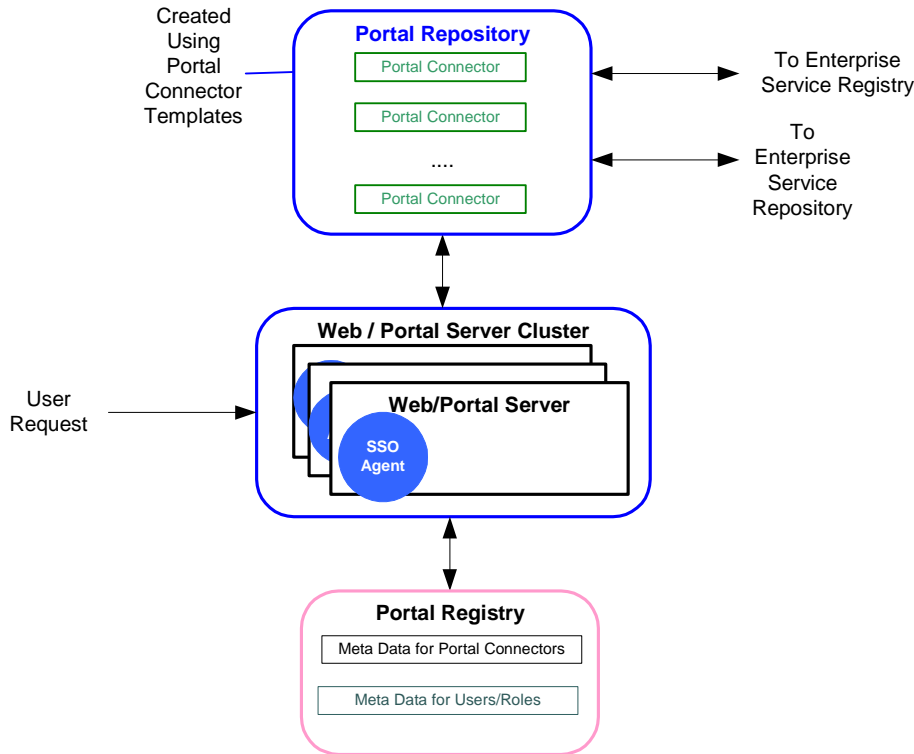
Figure 3-1: TFWeb Vision

The TFWeb Web Services architecture is a multi-tiered architecture consisting of a presentation tier (portal), an application tier (business logic), and a data tier. The proposed architecture is a comprehensive solution that provides a number of benefits, including:

- ÷! Multiple physically distributed portals but one logical portal.
- ÷! Load balancing and clustering will provide scalability and high availability.
- ÷! An open, flexible, distributed, and extensible architecture.
- ÷! Minimized risk of introducing new technologies.
- ÷! A strategic foundation for business and process integration to provide a consolidated view of the enterprise.

### 3.3 Components of the Portal

The portal layer consists of the portal engine, portal registry, portal repository and portal connectors. It provides the standard services to provide the aggregation of content. The content will be provided by making use of services provided by the application owners within the enterprise service repository.



**Figure 3-2: Portal Framework Components**

## 3.4 Portal Engine

The portal server will provide the standard set of services necessary to deliver presentation, personalization, customization, information access, integration and infrastructure.

The TFWeb portal will use the Computer Associate's (CA) Jasmine ii Portal 3.5 that runs as a Java Servlet and conforms to Java Servlet 2.2 and JSP 1.1. specification. The CA Jasmine Portal server can be installed in a single or multiple-server (i.e., cluster) configuration. Regardless of how it is installed at a given location, applications/services will operate, and be accessed, in the same manner since the portal repositories are replicated to all portal servers across the DoN Enterprise Portal.

### 3.4.1 Portal Runtime Environment

The CA Jasmine Portal will be staged on the following runtime environment.

#### 3.4.1.1 Servlet Engine

The CA Jasmine ii portal server will be staged on the BEA Weblogic Express 6.1 application server. This product is scalable and interfaces well with the service repository, as it is basically a scaled down version of the BEA Weblogic Application Server product. It supports SSL, JSP and Java Servlets, but does not include support for EJB and JMS, none of which is required for the CA Jasmine portal. BEA Weblogic Express will be installed on each Jasmine ii Portal server machine prior to installing the portal engine.

#### 3.4.1.2 Java Run Time Environment (JRE)

The Jasmine ii portal also requires Java Run Time Environment 1.3 (JRE 1.3) installed on each server running the portal engine prior to installing the portal engine.

### 3.4.1.3 Java Plug-in

The Transactions Applet in Portal Administration requires the Java Plug-in, version 1.2.2 or later installed prior to installing the portal engine. This plug-in is only required for workstations used by admin personnel who will be managing transactions.

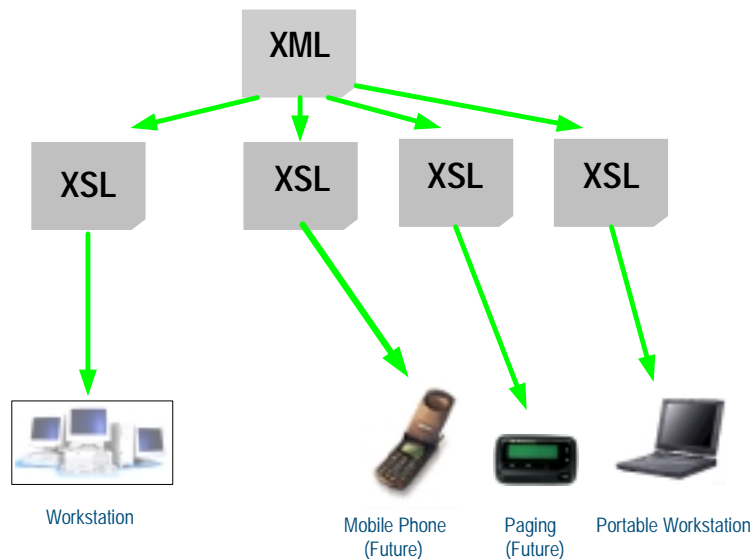
## 3.4.2 Portal Server Features

The following are some important key features provided by the portal server:

### 3.4.2.1 Presentation and Rendering

The portal presents information through the use of Work Places. Each Work Place directs the user to a web page within the portal, which contains one or more logical portal connectors that connect the user to the content.

The rendering of each page is dependent upon the device used to load the page, the role of the user connecting to that page and the data type being displayed. Dynamic rendering is achieved by applying XSL style sheets relevant to the device used.



**Figure 3-3: Presentation and Rendering**

XSL style sheets are used to define the access for each communication channel. Data is formatted in XML by portal compliant applications, or is transformed into XML format by the integration framework prior to portal access. The style sheets allow this common data to be formatted based upon user context.

A variety of physical devices can be used as web clients. The presentation layer of the application should be designed with display and user input device independence in mind, and are required to support the presentation standards supported by the Enterprise portal (e.g., appropriate D/HTML, XML, WML versions).

### 3.4.2.2 Personalization and Customization

Personalization allows users to change the look of their portal at multiple levels to suit their preferences. Users can apply themes to change the general look and feel of the portal (e.g. color, fonts and graphics). The content is arranged on the pages (WorkPlaces) as a collection of portal connectors. The user can select portal connectors from the available list and arrange them on the page if the user's access allows them to do so. The user can also remove, minimize or detach the connectors to improve viewing. The customization feature allows the portal



administrator to filter the content based on user workgroups and to push content to the users. It is important to note that the administrator has the ability to lock connectors to control what the user can customize.

Personalization and customization are achieved through an interface with directory services data. Users are assigned to workgroups based on several factors including their command and what types of tasks (or applications) they will be performing with the portal. In general, the rule is that users can see all applications available through the portal unless the visibility of that application is specifically restricted. If a user attempts to gain access to an application for which they have not been granted specific access, the portal service will display a form for the user to generate a request for access. User groups are also used to assign the template for look-and-feel based upon the needs of the organization to which they belong.

## 3.5 Web Browsers

The browser for use with the Enterprise portal must conform to open standards such as HTML/DHTML and XML. The browser must conform to the TFWeb Security Model (e.g., signed mobile code, 128-bit encryption). TFWeb will leverage off of NMCI and TFW Afloat to provide the browser-based technologies.

For the current version of the Enterprise portal, Microsoft Internet Explorer (version 5.5 or later) and Netscape (Professional Edition version 6.0 or later) browsers are acceptable as the lowest common denominators that are consistent with current Navy-wide browser requirements. Any service or application that is integrated into the Enterprise portal must be tested with and supported by these browsers. There may be an issue with configurations that currently support older versions of Microsoft IE or Netscape. Many of the current portal applications do not support older versions of Microsoft IE or Netscape (e.g., versions prior to Netscape Professional Edition v6.0 do not natively support XML). The primary functionality of an application must work correctly in the approved browsers, older versions will not be supported.

Since this is the main 'client storefront' (whether as an executable browser application or web-enabled desktop environment), there are certain key integration points that developers must concerned about.

### 3.5.1 Microsoft Internet Explorer

Microsoft Internet Explorer (IE) runs in the Macintosh, all recent MS Windows, Linux (using emulation), Solaris, and HP-UX operating environments. Depending on the operating environment, IE requires 16 to 64 MB of random access memory and 27 to 80 MB of hard disk space to operate properly.

Microsoft IE supports HTML, DHTML, XML, XSL, Java, Cascading Style Sheets (CSS) Level 1 and the Document Object Model (DOM) Level 1 standards.

### 3.5.2 Netscape

Netscape runs in the Macintosh, all recent Ms Windows, Solaris, Linux, HP-UX and most other UNIX operating environments. Netscape supports HTML, DHTML, XML, XSL, Java, Cascading Style Sheets (CSS) Level 1 and the Document Object Model (DOM) Level 1 standards in Professional Edition version 6.0.

### 3.5.3 Personal Digital Assistants (PDAs)

Use of PDAs is increasing as their capabilities mature. TFWeb will publish guidance that will enable application/content developers to simultaneously support users that employ PDAs for the following uses:

- Web Browsers (see above), understanding that the capabilities of PDAs are, by nature, less than that of a fully featured web browser. This will include they synchronization of more static web pages for data content or briefings, as well as certain 'news' content.



- ÷! Calendaring and Contact Management to synchronize with the approved office automation products.
- ÷! Basic E-mail support and file perusal, supporting the approved office automation product formats.

### 3.5.4 Cell Phones/Pagers

Wireless applications on cell phones include WML and Hand-Held Device Markup Language (HDML) browser-based standards, as well as the emerging Java 2 Micro Edition (J2ME) client software.

### 3.5.5 Handheld Scanners

Any specific information about support for handheld scanners will be documented in this section of the guidance document in the future.

### 3.5.6 Presentation Standards

Web content is typically derived from either static HyperText Markup Language (HTML) files stored on the web server, or from dynamic data, as in a database. Static HTML is easy to create but is difficult to maintain on large web sites because the look and feel of the web site is stored inseparably from the data. Best commercial software development practices dictate that the look and feel (presentation) should be separated from the data (content) thus allowing them to be managed separately. That is clearly the goal in moving towards application/data integration.

The consistent look-n-feel that will be required for the Enterprise portal is being defined by the DON CIO and will be documented in this section.

#### 3.5.6.1 Media Format Conventions

The TFWeb portal will support the following media formats (and versions, where applicable):

- ÷! Static Graphics formats
  - . ! JPEG
  - . ! GIF
  - . ! TIFF
  - . ! BMP
- ÷! Text/document formats
  - . ! ASCII
  - . ! PDF
  - . ! MS Office product formats (as natively supported by MSIE/office object inheritance)
- ÷! Markup Languages
  - . ! HTML
  - . ! XML/XSL
- ÷! Video/Animation formats
  - . ! MPEG
  - . ! Realplayer - RA
  - . ! Video for Windows - AVI
  - . ! Macromedia Flash - SWF

- . ! Quicktime Movie - MOV
- ÷! Audio formats
  - . ! Audio Interchange File Format - AIFF
  - . ! Microsoft Sound - WAV
- ÷! Compression types
  - . ! ZIP
  - . ! TAR
  - . ! UUencode

### 3.6 Portal Registry

The portal registry will house the list of the portal connectors with the associated metadata and information about users and roles. It will also house the personalization and customization information. The information in the registry will change when an administrator customizes the portal or users modify their preferences. APIs will be published to provide access to the registry. The registry will be replicated to support the multiple-physical and single-logical portal architectures.

The CA Jasmine portal requires a RDBMS to implement its portal registry. The portal servers in a cluster within the same physical portal will share the portal registry. The Portal registry will be replicated across the Portal Clusters, i.e. across multiple physical portals. The portal servlets communicate with the database using the appropriate JDBC driver.

The Portal registry will be implemented using Microsoft SQL Server 2000 for the first phase of TFWeb portal implementation. The Microsoft SQL Server 2000 database engine and database will reside on a server other than the one on which the Portal Server resides.

### 3.7 Portal Repository

The portal repository is a standard set of services that store portal configuration information such as the application connectors that provide connection to the enterprise service repository. The information in the portal repository will change when a new connector or content is published. The repository will be replicated to support the proposed multiple-physical and single-logical portal architectures.

The Portal repository is a flat file directory structure where the content will be published. The portal servers in a cluster will share the portal repository. The Portal repository will be implemented on a SAN (Storage Area Network) to provide scalability. The content can be published as a Universal Resource Locator (URL) or Universal Resource Identifier (URI) or a physical file. In order to conform to the TFWeb architecture the content will be published ONLY as a URI. Files in the repository are identified outside the portal as encrypted files using a sequential numbering scheme. All information about that content is stored in the registry, such as name and file type (mime type).

### 3.8 How does Portal connect to the Application/Data

The portal will connect to the applications/data using Portal Connectors. Portal connectors connect to the services in the enterprise repository, which in turn provides the connectivity with the applications/data, or local repositories.

#### 3.8.1 Portal Connector

The portal will connect to the services in the enterprise service repository using portal connectors. The services in turn provide the connectivity with the applications/data, or local service repositories. The portal connectors and services will conform to the portal to repository interface specification defined in the TFWeb Application and Information Services Architecture. The portal

connectors will provide the initial level of connectivity to the service repository to support the required levels of integration (as discussed in section 2).

The portal connectors (Components) are the presentation building blocks of Jasmine. They display content from a data source in the portal. The term Portal Connector is synonymous to Component, Gadget, CDA (Content Delivery agent) or Portlet.

In the TFWeb portal implementation all the portal connectors will be derived from a standard Portal Connector Template (PCT).

### 3.8.2 Portal Connectors vs. Services

The Portal Connector and the Service are NOT one and the same. The portal connector resides in the portal repository (which is specific to the portal product) whereas the services will reside in the Enterprise Service Repository, which is physically different than the portal repository.

The portal connectors can be seen, as in a thin layer of wrapper around the service. The portal connector will connect to the service via HTTPS request/reply from the portal server.

A portal administrator using administrative tools will publish portal connectors. The Enterprise Service Administrator will make the service available in the Enterprise Service Repository.

NOTE: There is a “one-to-one” relationship between services and portal connectors in that a portal connector can only display a single service. If an “aggregate” service is desired, (i.e., combining Service ‘A’ and Service ‘B’), a developer must create Service ‘C’ (with the logic to combine ‘A’ and ‘B’) which would then be published via a new portal connector.

For more information on Services please refer to section 4.

#### 3.8.2.1 Hyperlink Integration Requirements

The following are the minimum items that must be provided by the application developer in order to support Hyperlink Integration:

- ÷! Certified service module containing the hyperlink(s) to the application
- ÷! Service Registry Metadata Information (e.g., title, description, POC, keywords)
- ÷! Access Control List (ACL) to identify users who have visibility to the service

#### 3.8.2.2 Presentation Integration Requirements

The following are the minimum items that must be provided by the application developer in order to support Presentation Integration:

- ÷! Certified service module containing a hyperlink to the application, and appropriate descriptive information, or a re-direct
- ÷! Access Control List (ACL) to identify users who have visibility to the service
- ÷! Service Registry Metadata Information (e.g., title, description, POC, keywords)
- ÷! Any required XML Schema(s) and/or XSL Template(s) required for presentation rendering.

#### 3.8.2.3 Application/Data Integration Requirements

The following are the minimum items that must be provided in order to support Application/Data Integration:

- ÷! Certified service module that provides application/data connectivity
- ÷! Access Control List (ACL) to identify users who have visibility to the service
- ÷! Service Registry Metadata Information (e.g., title, description, POC, keywords)

÷! Any required XML Schema(s) and/or XSL Template(s) required for presentation rendering.

### 3.8.2.4 Access Control

A portal connector and a service handle access control differently. The TFWeb architecture will follow the “Access then Deny” approach, i.e., all portal connectors are visible to all users. Based on Department of Navy (DoN) policy, there may be a few exceptions for high security applications where the corresponding portal connectors will only be visible to a few authorized users. The majority of the portal connectors will be available to all the workgroups (roles) in the portal and access to the portal connectors is not role-based with some exceptions as stated above.

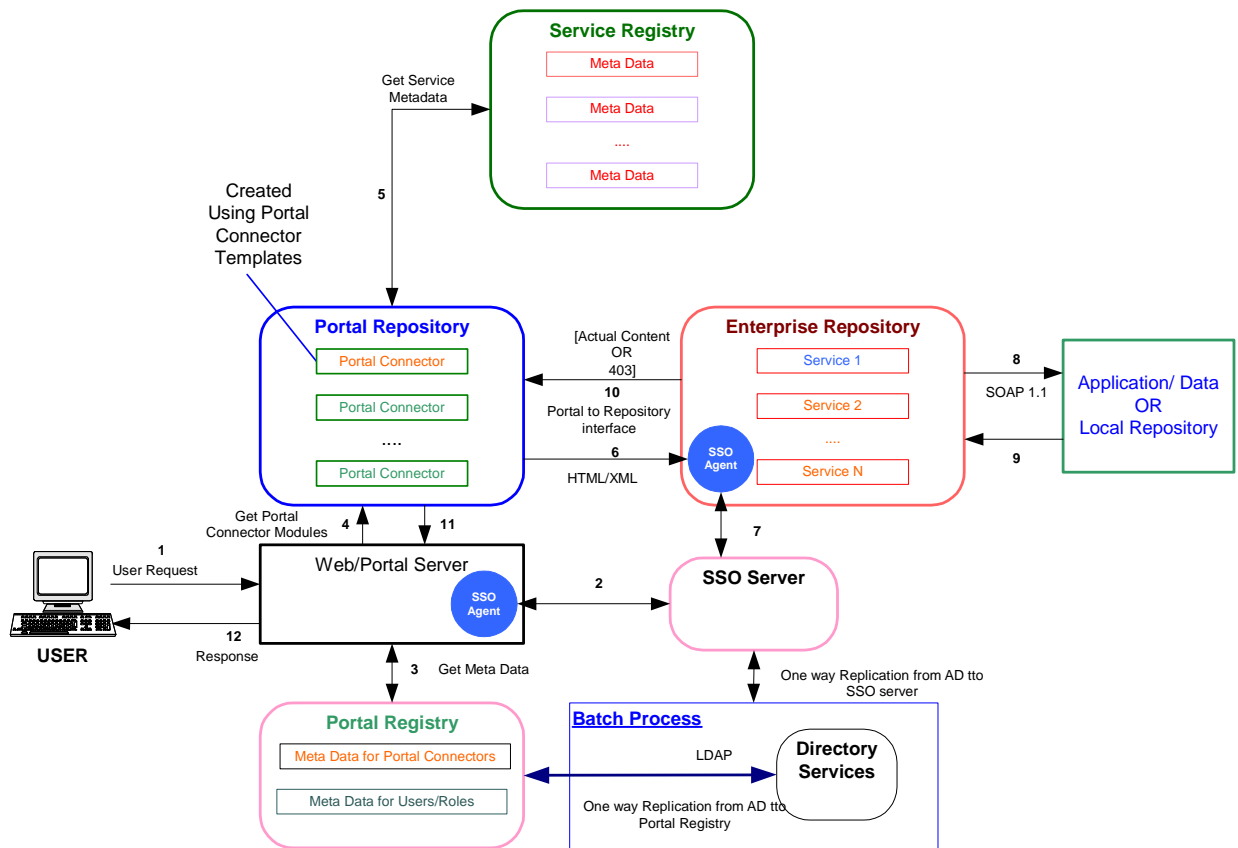
In the case of a service, the authorization is provided through the Universal Resource Identifiers (URI). It is the service owner's responsibility to provide the necessary authorization information to the Directory Services Administrator to set the access control list.

The users can arrange the portal connectors on their workspace of choice by selecting them from the ‘Library’ or ‘Channel’ section in the portal. The portal connector will try to connect to the underlying service by sending a HTTPS request to the service repository. The user's request will be authorized at the service level.

Thus in some cases, even though the portal connector is visible to the end user, the actual contents may not be available. The user will need to request access through the service owner.

Please refer to Section 6 for details regarding security and authentication.

## 3.9 Portal Operational Flow



**Figure 3-4: Portal Operational Flow**

The portal operational flow is as follows:

1. A user connects to the IP network through a *delivery channel* such as a network capable hand-held, notebook, or desktop computer. A delivery channel is the access device and network from which a user interacts with applications. For the first release of TFWeb portal, the desktop browser will be the ONLY delivery channel.
2. The Single Sign On broker will authenticate the user onto the Web Server and pass on the user request to the portal framework.
3. The portal framework will establish the *context* of the user request. The context is the identity and channel for the user request. A user may be authenticated at one or more levels of security, and a delivery channel may support some or all of these security levels.
4. The portal framework determines a user's workgroup and authorization. The user profile for the context is accessed to determine which templates and channels are available to the user. The portal presentation and rendering services determine the type of display method. Note: All portal connectors (except in certain SIPRNET instances) will be visible; However, if the user doesn't have authorization to a portal connector dragged onto one of their workplaces, they will see a form to request access instead of content.
5. The portal connectors will be based on a common portal connector template. The common template will basically perform the following activities and is common for all three types of integrations discussed earlier.
  - Get the metadata about the service from the Service Registry, e.g. integration type, link reconfigure flag, etc.
  - Collect all the necessary data required as per the Portal to Repository Interface specification.
  - The HTTPS request/reply mechanism will be used for all three types of integration.
    - i. Hyperlink Integration: A service will be created in the logical service repository that will respond with HTML/XML containing the Hyperlink(s). The portal to repository interface will provide 'flags' that the service developer can use to indicate if the links should be reconfigured or not. In this case the application is launched in a separate browser window so the service developer may choose not to reconfigure the links within the application to go through the portal. In this case it will be the responsibility of the application owner to provide the ACL for such links.
    - ii. Presentation Integration: Same as above except that the application is launched within the pane of the portal using an 'IFrame' and all communication between the user and the application will flow through the portal. In order to achieve this the links in the content will be reconfigured using the reverse proxy mechanism provided by the enterprise portal.
    - iii. Application/Data Integration: A service will be created in the logical service repository that will respond with the content in the form of HTML/XML. The portal to repository interface will provide 'flags' that the service developer can use to indicate if the links should be reconfigured or not. In this type of integration it is highly recommended to set the flag so that the links in the content will always be reconfigured to route every further request through the portal.

- Send a HTTPS request to the Repository along with the above data to connect to the Service.
  - The SSO and Directory Services Infrastructure will be leveraged to provide the ACL for the services for all three types of integration. The SSO plug in (agent) on the enterprise service repository will challenge the requests from the portal. The response will be either an access denied error (HTTP 403) or the actual output from the service.
  - In case of an access denied (403) error, a standard informative message indicating the contact information of the service/application owner will be displayed in the portal connector.
  - If the user is allowed access to the service, the service will execute and send a response back to the portal connector that will be either HTML or XML and an XSL style sheet.
  - If the content returned to the portal connector is XML with XSL style sheet, then the style sheet will be applied to convert to produce HTML on the server side.
  - If the 'reconfigure link' flag is set, then reconfigure all the links in the content to point back to the portal. In this case all the requests from the portal connector will be processed through the portal. In case of presentation integration and Application/data integration this flag will be ignored and the links will always be reconfigured.
  - Render the HTML content in the portal connector.
6. All the portal connectors on the requested page/work place will be executed as stated above.
  7. The portal engine waits for all the portal connectors to execute and receive the data. The collective data is formatted for presentation to the user based on their delivery channel. Even though one or more application connectors fail to execute or do not receive data in the specified time, the portal framework proceeds further to render the data as mentioned above.

### 3.10 Portal Development Kit (PDK)

The TFWeb Portal Infrastructure provides a portal development kit (PDK). PDK is a set of tools to help the Portal Developer and Service Developer to develop the Portal Connectors and Services respectively. PDK consists of three tools:

1. Portal Connector Template
2. Portal Connector Stub
3. Application/Service Help

#### 3.10.1 Portal Connector Template (PCT)

A Tool for the Portal Developer to help develop a portal connector very rapidly but conforming to a standard template and also conforming to the Portal to Repository Interface Specification.

The Portal Connector Template is a standard code template consisting of a set of APIs that need to be in the PCT in order to connect from the portal to the Repository and vice versa. The PCT will support all the three types of integration discussed above.

Once the a service (developed by a service developer) is certified and available in the Enterprise Repository and registered in the Enterprise Service Registry, the portal developer/administrator will use the PCT to create a portal connector for a service. The administrator will then publish the portal connector in the portal to make the service available in the portal.

### 3.10.1.1 Pseudo Code for the Portal Connector Template (PCT)

Following is an example of how the Portal Connector template may look. This should be used as guidelines and should not restrict the engineering / design effort. The portal template will be implemented using JSP and Java Servlet technology. Most of the logic will be in the Java Servlet allowing the portal connector template be a very light JSP.

1. Execute Portal Connector (PC)
  - a. Get Service Metadata (Owner, URL etc.) from the service registry. The service URL will be used only for the first time to connect to the service.
  - b. Get 'source data' required as per the PRI specification and package it into the XML format.
  - c. Set the time out (attribute in the Service Registry) for the HTTPS request.
  - d. Send an HTTPS request to the Repository. The mechanism will pass the standard HTTPS request header from the user request (along with the cookies etc) to the repository. Similarly pass the standard HTTPS response header received from the repository back to the user's request channel (browser). During this process it will not loose/drop any information from user's request and repository.
  - e. If the response is not empty
    - i. If error then display the corresponding form
    - ii. Else start processing the content returned as response by the URI (service) as below.
  - f. If response is not empty AND No Error
    - i. If Integration type (One of the attributes from the PRI data) is 'Level 3' (Application/Data Integration) AND if the response is an XML with XSL style sheet then use APIs to apply the style sheet.
    - ii. If the 'Reconfigure Links' flag (One of the attributes from the PRI data) is 'TRUE' then call APIs to reconfigure the link(s) in the 'response' from the service.
    - iii. Append the portal connector template specific standard code to the formatted response to make sure that the standard code will be available for the next request.
2. Render the content in the portal connector.

The portal connector template will be made available in the time frame when the Beta Labs will be made available to the service developers to integrate and test the services in the model office environment (Beta Lab).

At that point of time the portal developer in the Beta Lab will use the Portal Connector template to create the Portal Connector corresponding to the service to be tested and conduct the testing as per the test plan.

### 3.10.2 Portal Connector Stub (PCS)

Portal Connector Stub (PCS) comprises of simple web pages that simulate the behavior of the Portal Connector Template. It allows the application developers to test out the service interfaces without needing to install the Portal Server at their development sites.

Developers will use a web server and the PCS at their site. The application owners develop and test their application interfaces with PCS following which they submit the service for integration testing and certification.



The PCS is developed using Hyper Text Markup Language (HTML), JavaScript and Active Server Pages (ASP). This tool allows the service developer to test the service by simply typing in the service URL in the provided text box and launching the service. PCS passes the necessary PRI (Portal to Repository Interface) data in the Request Header sent to the Service. The service will then extract the information from the Request Header variable PRIData and use it appropriately. The HTML or eXtensible Markup Language (XML)+ eXtensible Stylesheet Language (XSL) returned by the service will then be rendered in the frame inside the PCS tool.

PCS is a 'test' stub' and not the real Portal Connector. PCS's purpose is to allow the Service Developer's to test the service without being required to install CA Jasmine Portal Server. Portal Connector Template is used at to create the actual 'Portal Connectors' for the portal to make the service available in the portal. In Portal Connector Template we will be certainly converting the XML+XSL to HTML on the 'server side'. To keep the PCS simple, it will rely on the browser's capability for the XML to HTML conversion. Thus the browser used for testing the service MUST be able to handle XML.

For more details please refer to section 11.

### **3.10.3 Application/Service Help**

This section will document the facilities that will be available for portal application/service owners to be able to submit and update Help information relating to their particular applications. The goal is to provide a single authoritative source of Help information for each of the applications that will be available in the portal, while minimizing the workload for portal administrators.

#### **3.10.3.1 Application Help Facilities**

The portal will provide application owners with a component that will allow them to enter and update Help documentation for their applications. This component will only be available to users with the proper role of Application Owner. There must also be some correlation between the application owner's user id and the particular application that he/she owns. This would prevent an application owner from modifying Help content for an application that he/she does not own.

The Help component must provide several mechanisms for the application owner to enter their Help content. One mechanism would be a text area that would allow the application owner to enter straight text about their application. Another mechanism would allow the application owner to enter a fully qualified URL that would point to a custom Help page that would be served from within their application domain (i.e. from a Web or Application server). This would allow for more dynamic content, including images, diagrams and screen shots.

The application Help information will be stored in a central portal database and replicated to all other databases in the Enterprise.

## **3.11 Portal Look and Feel**

The purpose of this section is to provide an overview of the TFWeb portal presentation layer. The portal presentation layer will have a consistent look and feel among both ashore and afloat implementations. It will consist of three major sections: the Header section, the Content section and the Footer section. The Header section will occupy approximately 15% of the browser, the Content section will occupy approximately 80% and the Footer section will occupy approximately 5%, from top to bottom. For the WEN Beta and Pilot implementation, the portal engine used will be Computer Associates Jasmine 3.5. CA Jasmine provides the ability for portal administrators to create many different templates, which control the look and feel of the portal. For the WEN Beta and Pilot implementation, several "TFWeb Standard" templates will be created for participating Commands and User Groups. All the templates will have the same components and color schemes, however they will differ in areas of Command Branding (images and artwork) and dimensions. In order to preserve a common look and feel across the TFWeb system, new templates shall only be created by copying "TFWeb Standard" templates and modifying the artwork and text in the designated branding areas.



### 3.11.1 Description

The purpose of the portal presentation layer is to provide the user with a common look and feel for all the components, applications and services that are available in the portal, as well as consistent navigational controls. This section will present the granular details of how common components of the presentation layer should look and behave in order to ensure a consistent and positive user experience on both ashore and afloat implementations.

### 3.11.2 Portal Layout Definition

The overall portal layout will be defined by a frameset file called "frameset.acs". Each template will have its own frameset file, which will define a Frameset of 3 rows. Each frame will contain a URL to the source file for the Header, the initial Content and the Footer (html, jsp). Color schemes and font styles for each template will be defined in a Cascading Style Sheet file: styles.css. Each template frameset, and its associated style sheets, will be developed and distributed by the TFWeb Portal Team.

In order to let Application Web Services use the same style sheets (CSS files) used by the portal templates, application owners must include the following line of code in their XSL documents:

```
"<LINK REL='stylesheet' HREF='/servlet/media/templates/' & <ClientStyle> & '/styles.css'
TYPE='text/css' title='TEMP_STYLES'>"
```

<ClientStyle> will be replaced with the "template key" corresponding to the user's selected template and that will guarantee the proper Style Sheet will be applied to the Application Services registered in the portal.

### 3.11.3 Header Definition

The portal header, also referred to as the banner, is divided into two basic regions:

- ÷! Branding for Command / User Group
- ÷! Portal navigational controls

The header should support varying browser dimensions, with a minimum of 800 x 600 pixels. The right side of the header must stretch for larger browser windows, up to a maximum of 1280 x 1024 pixels. The header must contain the following pieces of information, graphics and controls:

- ÷! Command / User Group branding / graphics
- ÷! The title "Navy Enterprise Portal"
- ÷! Portal navigational buttons / links
- ÷! Portal library search capability, including an execute button / link

#### 3.11.3.1 Branding for Command / User Group

The branding region of the header will allow for customization based on a particular Command or User Group. This region is where relevant graphics and artwork will be displayed. For example, a header for the USS George Washington CVN 73 Command could contain images of the carrier and its supporting battle group. See the "Header Development" section below for more details.

#### 3.11.3.2 Portal Navigational Controls

The portal navigational controls region will contain techniques for the user to navigate to the different areas of the portal. These controls may be in the form of graphically represented buttons. The colors on the controls shall clearly indicate which "mode" the user is in. Color shall not be the only control indicator, but provide a graphical indicator as well, such as a raised button, outlined button, or other graphical representation that would differentiate the current "mode" from all other "modes". It is suggested that each control have some type of onmouseover / onmouseout behavior that would highlight the control as the cursor passes over it and then revert back to its original state once the cursor has left the control.

There will be four major navigational controls. The spelling and uppercase / lowercase letters shall not deviate from the list below:

- ÷! Home
- ÷! WorkPlaces
- ÷! InfoStore
- ÷! MyProfile

The left to right order of these buttons shall be Home, WorkPlaces, InfoStore, MyProfile.

There will also be two minor navigational controls that will be represented by plain text or graphical text images. The spelling and uppercase / lowercase letters shall not deviate from the list below:

- ÷! Help
- ÷! Logoff

The left to right order of these controls shall be Help, Logoff.

The Navigational controls are implemented in the "TFWeb Standard" templates and shall not be modified.

#### **3.11.3.2.1 Home**

The Home control will be initially selected (turned on) when the user logs into the portal. All other controls will not be selected (turned off). In addition, the Content section of the portal will contain default components, aggregated into what will be known as the Home page. These components will be available to all users regardless of Role / Workgroup. Additional information regarding the Home page is defined in a later section of this document.

If a user is viewing / working other areas of the portal, such as WorkPlaces, he can click on the Home control to return to the Home page.

#### **3.11.3.2.2 WorkPlaces**

The WorkPlaces control will allow the user to navigate to the WorkPlaces area of the portal. Clicking on the WorkPlaces control will cause this control to be selected (turned on), while all other controls will not be selected (turned off), and the Content section will be replaced by the user's default WorkPlace and the WorkPlaces toolbar which allows access to all the user's WorkPlaces. The HTML code behind this button shall be an anchor tag that invokes the server process to populate the Content section:

```
<A HREF="/servlet/portal/workplace">WORKPLACES CONTROL </A>
```

#### **3.11.3.2.3 InfoStore**

The InfoStore control will allow the user to navigate to the InfoStore area of the portal. Clicking on the InfoStore control will cause this control to be selected (turned on), while all other controls will not be selected (turned off), and the Content section will be replaced by the InfoStore components. The HTML code behind this button shall be an anchor tag that invokes the server process to populate the Content section:

```
<A HREF="/servlet/portal/explorer">INFOSTORE CONTROL </A>
```

#### **3.11.3.2.4 MyProfile**

The MyProfile control will allow the user to navigate to the MyProfile area of the portal. Clicking on the MyProfile control will cause this control to be selected (turned on), while all other controls will not be selected (turned off), and the Content section will be replaced by the MyProfile

components. The HTML code behind this button shall be an anchor tag that invokes the server process to populate the Content section:

```
<A HREF="/servlet/portal/profile">MYPROFILE CONTROL </A>
```

### 3.11.3.2.5 Help

The Help control will display Portal User Help in the Content section. Clicking on the Help control will cause all other controls to set to their non-selected state (turned off). The HTML code behind this control shall be an anchor tag that invokes the server process to populate the Content section:

```
<A HREF="/servlet/portal/help">HELP CONTROL </A>
```

### 3.11.3.2.6 Logoff

The Logoff control will log the user out of the portal and present the user with a page containing several options:

- ÷! Log back into the portal
- ÷! Close the browser
- ÷! Links to TFWeb, NMCI, and other Navy / Marine Corps / DoD websites

The HTML code behind this control shall be an anchor tag that invokes the server process to log the user out of the portal and then displays the logoff option page detailed above:

```
<A HREF="/servlet/portal/logout">LOGOFF CONTROL </A>
```

### 3.11.3.3 Search Portal Library

The Header will contain a control that will allow the user to perform a search of the portal library at any time. This control shall consist of 3 parts:

- ÷! A text label entitled "Search", to designate what the control is
- ÷! An HTML Form element of INPUT TYPE="text", for user input, with a size of 20
- ÷! A button control or image that will execute the search

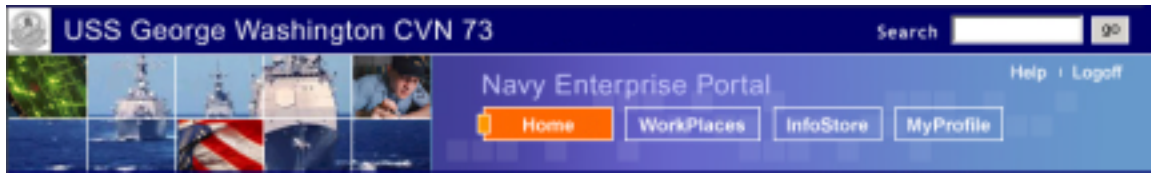
Upon clicking the search execute control, the Form containing the INPUT TYPE="text" element shall be submitted to the server for processing using the action "/servlet/portal/search/execute", with a submit method of "POST". The search results will be displayed in the Content section.

### 3.11.3.4 Header Example

The following figure is an example header, showing all regions and controls outlined above.



Figure 3-5: Portal Header Design Template



**Figure 3-6: Portal Header Example**

### **3.11.3.5 Header Development**

New template headers for Commands and User Groups for the WEN Beta and Pilot implementation can be developed by modifying TFWeb Standard templates using the procedure detailed in the following sections. In general, Commands will have the ability to replace the header's standard branding images with their own images. The images that can be replaced, depending upon the choice of TFWeb Standard template used, are as follows: Command Logo, Command Name, and Montage Branding.

Each Portal Administrator is ultimately the one who can import the final Jasmine ii "Templates" and assign them to workgroups. TFWeb policies address the procedures for approval and dissemination of new templates. Sample portal headers from Standard templates can be found in the following sections.

### 3.11.3.6 Portal Header Templates



Figure 3-7: Template Examples

### 3.11.3.7 Procedure to Create New Portal Templates

There are three main steps involved in the creation of new Jasmine ii Portal templates:

- Import an existing TFWeb Standard template.
- Select the template as your "Personal Template" so you can see the changes as you make them.
- Modify the imported template's image components.

Note: You must be a "Group Admin" or "Portal (User/Group) Administrator" to see these controls in the Portal.

**A. Import an existing TFWeb Standard template:**

1. Log into the Portal and click the "MyProfile" button. (Opens the "Profile" page)
2. Click the "Template Administration" link at the top of the page. (Opens the "Template Editor" page)
3. Click the "Edit Template" button. (A new page will appear) Note: Selecting a particular template from the drop down menu will not effect the end result for this process, unless you make a mistake at step 7!
4. At the top of the new page, click the "Import" button to open the Import page.
5. On the Import page, Click on the "Browse" button next to the "Import File" text field.
6. Select the TFWeb Standard Template from your local hard disk. (Previously downloaded from the Portal Library: Administration/TFWeb StandardTemplates)
7. Import Action: IMPORTANT! Click the radio button next to "Create a new template", NOT the default choice that will OVERWRITE EVERYTHING the currently selected template.
8. Click the "Import" button at the bottom of the page. ("Import completed successfully" should appear on a new Template Editor page showing all the template parameters).
9. Enter the desired name in the "Template Name" field
10. Click the "Save" button at the top of the page. ("Your template has been saved" should appear on the Template Editor page)
11. Import is now complete. See the next section to select and use the new template.

**B. Select and use the new Template in the Portal:**

1. Click the "MyProfile" button. (Opens the "Profile" page).
2. Select the newly created template from the "Personal Template" drop-down list. (It will be the last one on the list.)
3. Scroll to the bottom of the page and click the "Save" button. (Immediately, your Portal will reload and be updated to reflect the change, using the new template).

**C. Modify Template Images:**

**Note:** This step requires access to the Jasmine ii Portal server and write permissions in the following directory:

\\[server name]\portalserver\content\templates\[template#]\images\hdrNav

1. Open the hdrNav directory. Photoshop (PSD) files have been provided with the Template for branding and command identification. These files have been created and customized to precise dimensions according to specific requirements of the template in that directory. Modification of these dimensions will produce unwanted results!
2. Open the PSD files in Photoshop.
3. Open your command logos and images to be merged with the Template files.
4. Drag or Paste your command logo and images into their corresponding image layers for custom branding. NOTE: Leave the original images in the Photoshop file. If desired, "turn off" the Layer of the original image to avoid any unwanted overlapping.
5. Make your desired changes.
6. In Photoshop, select "Save for Web" from the "File" menu and save the file as either a .jpg or .gif in the hdrNav directory.
7. Test the changes by "reloading/refreshing" the browser.

8. When the changes are completed, the modified template can be exported (to a zip file) for possible approval and dissemination.

### 3.11.3.8 Content Definition

The Content section is the dynamic area of the portal that displays the various content, depending on the user's navigation. The Look and Feel of the Content section is defined by the Cascading Style Sheets for a particular template. These style sheets define the color schemes and font styles for each template. Style sheet classes are applied to various sections of the Content areas to provide a consistent look and feel throughout all pages and components of the portal.

### 3.11.3.9 Home Page

The Home Page will be the default page that will be displayed for every user when they initially log in. The contents of this page should be generic such that all users, no matter what role or workgroup they belong to, will have a consistent view of its content. Suggestions for inclusion on this page are:

- ÷! Portal tutorials
- ÷! Navy / Marine Corps / DoD / Government news links
- ÷! Navy / Marine Corps reference links

The exact content of the Home Page has not been determined at this point in time. This document will be updated to include the contents of this page once it has been defined.

The Home Page will also include a welcome message to the user in the upper left corner of the page. The message will display their name and, if they are military personnel, their rank or rate. The format of the message will be:

- ÷! Military personnel: Welcome <rank>|<rate> <firstname> <lastname>
- ÷! Civilian personnel: Welcome <firstname> <lastname>

The data for the user's first and last name as well as their rank or rate will come from the portal's user repository.

### 3.11.4 Footer Definition

The Footer section is the area at the bottom of the browser that is always displayed, no matter what is displayed in the Content section. The Footer is divided into three basic regions:

- ÷! Display any important messages that should be viewed by the entire portal user community
- ÷! INFOCON status, which alerts portal users to any kind of threats to the NMCI / WEN network infrastructure and actions users should take to protect DoN resources
- ÷! Mandatory Government and / or DoD links



Figure 3-8: Footer Example

NOTE: These are NMCI requirements and may or may not be adopted in the TFW portal.



### 3.11.5 Portal Example

The following figure is an example portal, showing all sections as detailed above:

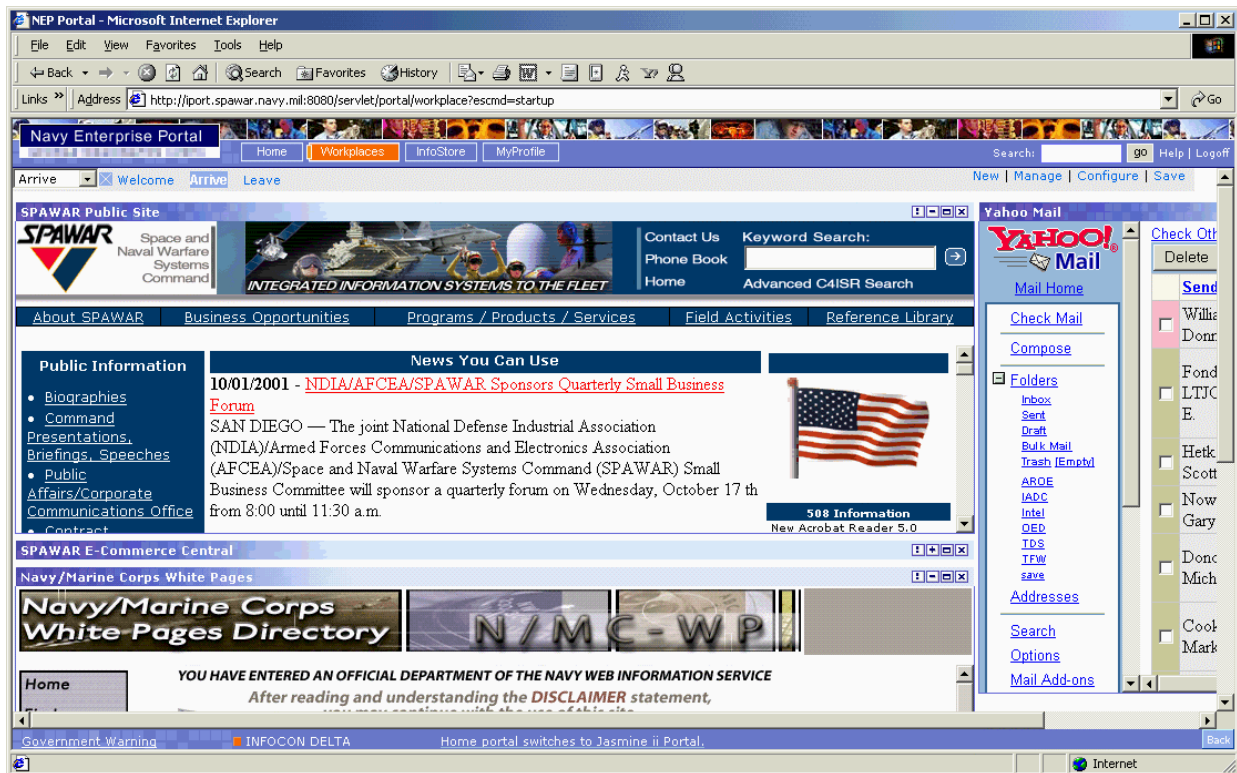


Figure 3-9: Portal Example

### 3.12 Standards

The services and portal connectors MUST conform/leverage the following standards/technologies:

- ⇨ The Portal Connectors shall use the standard Portal Connector Template
- ⇨ The portal connector and the Service shall conform to the Portal to Repository Interface (PRI) specification.
- ⇨ Portal shall support Java Servlet 2.2 and JSP 1.1
- ⇨ Portal shall leverage TFWeb IA infrastructure.
- ⇨ The response from the service shall conform to W3C HTML 4.0 and XML 1.0 specification
- ⇨ The response generated by the service shall conform to Section 508
- ⇨ Portal shall display a link to the privacy and security policy.



## 3.13 Administration and Management

CA Jasmine provides fully web based administration tools for the portal server, users, and roles etc. management.

### 3.13.1 Portal Administration

CA Jasmine portal server provides a set of web-based tools for setting up, configuring, monitoring, and tuning the Portal Server and Clients. Only members of the Admin workgroup have access to Portal Administration. Following are a few important functions that the administrator can view / manage / perform.

- ÷! Statistics / Transactions
- ÷! System Messages and Transactions
- ÷! User Administration / LDAP synchronization
- ÷! Workgroup (Role) Administration
- ÷! Workplace Administration
- ÷! Template Administration
- ÷! Knowledge (content) Administration / publishing.
- ÷! Channel Administration.
- ÷! Manage Local / Global properties
- ÷! Import/Export portal Configuration

**Portal Administration**

- Statistics
- System Messages
- Transactions Applet
- User Administration
  - Current User Activity
  - Manage Users
  - Manage Workgroups
  - Default Workplace
- Object Management
  - Object Attributes
  - Preferences Configuration
- Publishers
  - Client-side
  - Server-side
  - Object Availability
- Advanced
  - Supported Languages
  - Connection Status
  - Content Types Editor
  - Update Portal
  - Proxy Configuration
- XML-API
  - XML-API Test
  - View request.dtd
  - View response.dtd
- Properties
  - Content Handlers
  - CSV

**Statistics**

Portal *ushrseims602:8080* started on: September 19, 2001 11:43:15 AM CDT

|                         |                   |
|-------------------------|-------------------|
| Total Hits:             | 225               |
| Documents Served:       | 59                |
| Connection Pool Resets: | 0                 |
| Database Errors:        | 0                 |
| Portal Version:         | 3.03, 20010823.01 |

**Currently Scheduled Classes**

| Name               | Description                                    | Interval (minutes) | Last Time Run    |
|--------------------|--|--------------------|------------------|
| VerityQueue        | Flushes pending search engine index operations | 1                  | 9/19/01 4:54 PM  |
| VeritySearchEngine | Optimizes the search engine index              | 720                | 9/19/01 11:54 AM |
| TaskMonitor        | Runs tasks scheduled in the POR_TASK table     | 1                  | 9/19/01 4:54 PM  |

Figure 3-10: The Jasmine portal Administration

### 3.13.2 Theme (Template) Administration

CA Jasmine portal's appearance / look and feel can be controlled using the Templates. New templates will be from scratch or by modeling them on existing templates. Only the users who are members of the Admin or Template Editors workgroups can edit templates.

Features controlled by the template include:

- ÷! Page layout
- ÷! Colors of various parts of the window
- ÷! Custom images to enhance the meaning and visual effects of the user window
- ÷! Button images
- ÷! Font style and size
- ÷! Size and placement of separate framed areas (if frames are used).

Content for templates resides in two locations. User access and customizable features such as fonts and colors, reside in the database. The html files and other related files are stored in the templates directory, which is a sub-directory to the previously mentioned content directory. These files can be managed through any web site editor. They contain the frame sizes and all reference to files needed to display the border area of the template's window. These files are exported from and imported into the portal using the "ZIP" format.

Following is the screen shot of the template editor.

---

Save

Cancel

Import

Reload

Reset

Template Key:

15

Template Directory:

D:/CAportalserver/content/templates/15/

Template File:

[D:/CAportalserver/content/templates/15/main.acs](#)

Template Name:

EDS

Framed Template:

No: ☐ Yes: ☒


Frameset Name:

[D:/CAportalserver/content/templates/15/frameset.acs](#)



Cached Template:

No: ☐ Yes: ☒

Locale:

English 

Description:

EDS  
Framed Template   


### Workplace Configuration

|                            |                      |
|----------------------------|----------------------|
| Background Color:          | <input type="text"/> |
| Workplace Highlight Color: | <input type="text"/> |
| Element Background Color:  | <input type="text"/> |
| Tab Color:                 | <input type="text"/> |
| Property Image:            | <input type="text"/> |
| Minimize Image:            | <input type="text"/> |

Figure 3-11: The Jasmine template Administration

### 3.13.3 Portal Connector Development

The portal server is used for the role based presentation and rendering. The business logic will be handled by the services in the repository. As far as the portal is concerned very little development effort is required due to the introduction of 'Portal Connector Template'. The CA Jasmine portal does not provide any development environment to develop or debug the portal connectors. Developers can use an environment of their choice.

## 3.14 Roles / Responsibilities of a Portal Administrator

Some of the important roles and responsibilities of an administrator are as follows.

### 3.14.1 Workgroup (Role) Administration

Define workgroups for the portal users based on the user groups in the Active Directory and the functional roles within the DON organization.

### 3.14.2 User Administration

Import users from the Active Directory into the portal's user store. Verify that the user is assigned to the proper workgroups. To support the MAC (Move/Add/Change) the portal user store **MUST** be kept in synch with the Active Directory. An Active Directory Synch utility will be provided to the administrator.

Individual users can be de-activated without removing them from the portal store. This can be used to temporarily deactivate the user account if necessary.

### 3.14.3 Template Administration

The workspace administrator will create and maintain the templates. The administrator will also maintain the access control to the templates by assigning the templates to the users or workgroups.

### 3.14.4 Knowledge (content) administration / publishing.

Once the portal connector is certified, the InfoStore administrator will publish the new portal connectors into the portal. The administrator will also maintain the access control to the portal connectors by assigning the templates to the users or workgroups.

### 3.14.5 Channel Administration.

The channel administrator will create and maintain the channel. The administrator will also maintain the access control to the channel by assigning the channel to the users or workgroups.

### 3.14.6 NMCI services administration

The ISF-NMCI Administrator will be provided access to the NMCI services like ExecuView, Remedy, Tivoli etc. applications/services, if a web front end is available for these applications.

## 3.15 Roles / Responsibilities of a Portal Developer

Here are a few important roles / responsibilities of a developer.

### 3.15.1 Develop Themes

Develop templates (Themes) to be assigned to the roles/workgroups. Themes define the style sheets (color schemes and font schemes) and the general look and feel of the portal i.e. the images to be used for the tabs, logo, background colors etc. Developers should only duplicate and modify images (branding) of existing templates, not create entirely new templates which could depart from the TFW look and feel guidelines.

### 3.15.2 Incorporate Portal Templates with Services

This section illustrates issues to be considered by the Service Developers for the incorporation of portal-defined templates and styles. The portal uses different style sheets for each template or theme; however, the name of the style sheets and their elements (referred to as “tags”) remain the same.

It is advisable that the Service Developers incorporate these cascading style sheets (CSS) into their applications to keep the look and feel across all applications consistent with the portal and to create a more seamless, user-friendly experience. In some instances, maintaining a template's predefined color palette may be critical for a particular working environment, such as a ship's command center where the implemented template may be designed for a dark room environment and a bright white application would be hard to use. Incorporating the CSS will promote display consistency across multiple pages and once incorporated, will save time in both developing and maintaining existing and new applications.

How to incorporate the Portal Template into Service:

1. Include the URL path used to reference the Cascading Style Sheets
2. Include tags (selectors/elements) in applications to define the attributes to be implemented

#### 3.15.2.1 Include the URL path used to reference the Cascading Style Sheets

The Service Developer must reference the style sheets at the top of their web pages, between the “header tags”.

For example:

```
<HEAD>
<LINK REL='stylesheet' HREF='/servlet/media/templates/' & <ClientStyle> & '/styles.css'
TYPE='text/css' title='TEMP_STYLES'>
</HEAD>
```

The above referenced style tag will be an attribute defined in the PRIDataRequest as ClientStyle. If the Service Developer chooses to incorporate the user's look and feel into their application, the line above will need to be included in each page.

#### 3.15.2.2 Include tags (selectors/elements) in applications to define the attributes to be implemented

In addition to referencing the user's specific CSS, the Service Developer will also need to reference each style tag (Elements) as defined in the CSS.

For instance, below is HTML that may be in the Service Developer's current web application:

```
<Table>
  <tr>
    <td><b><font size="10" type="Arial" color="blue">Welcome, John Doe!</font></b></td>
  </tr>
</Table>
```

Assuming the CSS is referenced at the top of the page, the above would be replaced with:

```
<td class="welcome">Welcome, John Doe!</td>
```

where CSS tag “welcome” is defined in the stylesheet as:

```
.welcome
{
    COLOR: blue;
    FONT-FAMILY: arial, verdana, helvetica;
    FONT-WEIGHT: bold;
    FONT-SIZE: 10px
}
```

The value in using the CSS, is that if changes to the rendering are needed, e.g. fonts, colors, margins, typefaces and other aspects of style, on a web application, only one change is made in the CSS, rather than in all pages that use these styles.

How to reference specific the style tags defined by the CSS:

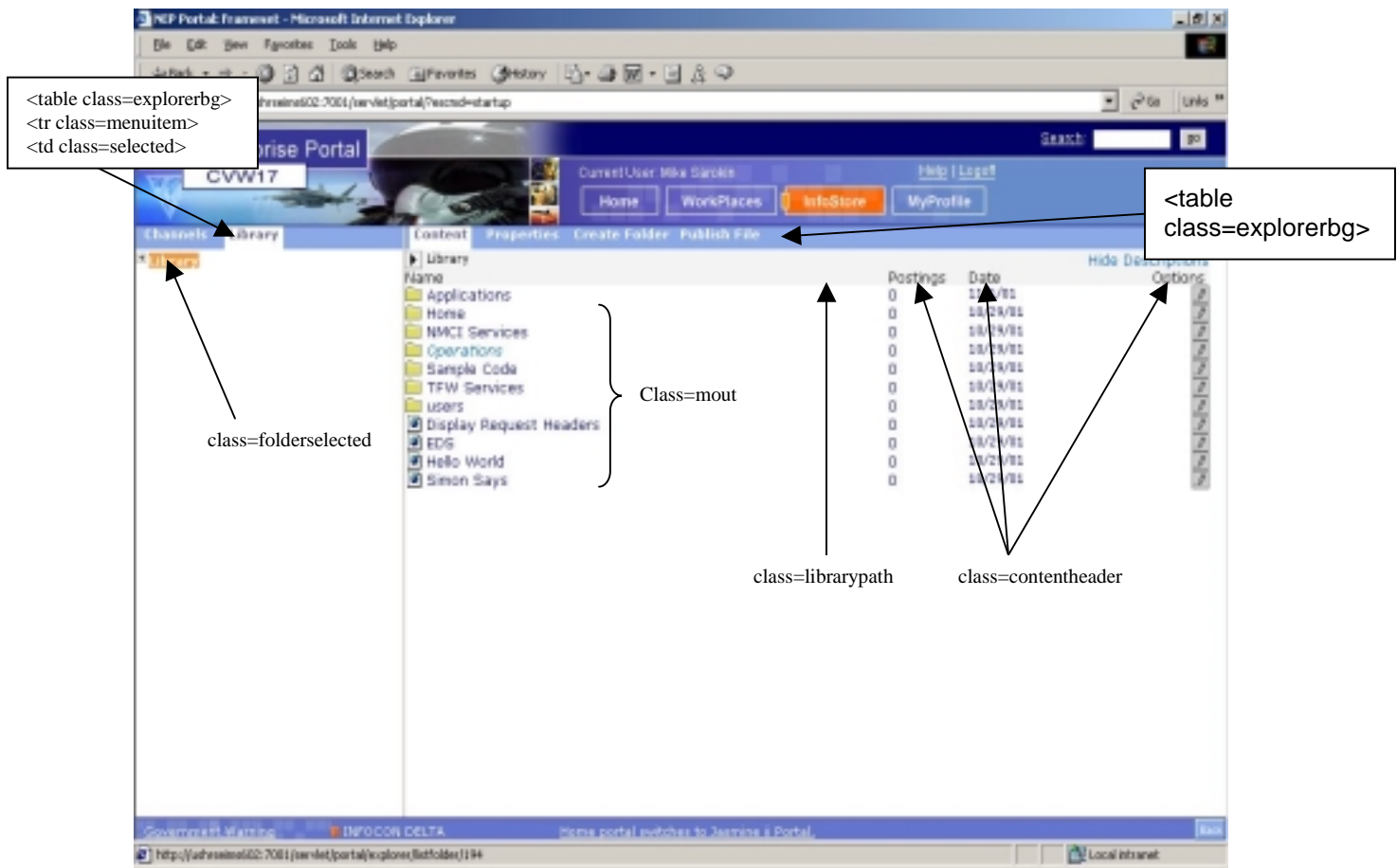
Below are a table and screenshots that demonstrate how to implement the style tags in the CSS. The table lists all of the elements and classes as defined by all style sheets used in the portal. The attributes for these elements and classes will change depending on the template chosen, however the code will not need to be modified once classes are referenced. The attributes listed below are one example of a template available on the portal. The screenshots map out where these have been used in a template for the portal. These may be used as a guideline for Service Developers to reference when adding the class names to their web applications.

**Table 3-12: Style Tag Descriptions for Style Sheets**

| Element/Class        | Description             | Attributes  | How to reference             |
|----------------------|-------------------------|---|------------------------------|
| <b>A</b>             | <b>Hyperlink</b>        | COLOR: #0163e4;<br>FONT: 10pt univers, verdana, arial, helvetica, sans-serif;<br>TEXT-DECORATION: none            | No additional code needed    |
| td                   | Table Data              | FONT: 8pt univers, verdana, arial, helvetica, sans-serif  | No additional code needed    |
| <b>th</b>            | <b>Table Header</b>     | FONT: bold 10pt univers, verdana, arial, helvetica, sans-serif  | No additional code needed    |
| contentheader        | Header                  | COLOR: black;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 12px                              | class="contentheader"        |
| currentdirectory     |                         | COLOR: #a9a9a9  |                              |
| explorerbg           | <b>Background Color</b> | BACKGROUND-COLOR: #8CAAE7   | class="explorerbg"           |
| explorertabindicator |                         | FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 12px;<br>TEXT-DECORATION: none                     | class="explorertabindicator" |
| explorertablebg      |                         | BACKGROUND-COLOR: #f3f3f3   | class="explorertablebg"      |
| file                 | <b>File font</b>        | COLOR: darkblue;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 11px                           | class="file"                 |
| fileselected         | <b>Selected File</b>    | BACKGROUND-COLOR: #FF9933; COLOR: #ffffff;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 11px | class="fileselected"         |
| folder               | <b>Folder Name</b>      | COLOR: #666699;<br>FONT-FAMILY: univers, verdana,   | class="folder"               |

| Element/Class   | Description                | Attributes  | How to reference    |
|-----------------|----------------------------|---|---------------------|
|                 |                            | arial, helvetica;<br>FONT-SIZE: 11px;<br>FONT-WEIGHT: bold  |                     |
| folderselected  | Selected Folder            | BACKGROUND-COLOR:<br>#FF9933;   | COLOR: #ffffff;     |
| font1           | Font option                | FONT: 12pt univers, verdana,<br>arial, helvetica, sans-serif  | class="font1"       |
| font2           | Font option                | COLOR: #919191;<br>FONT: 14pt univers, verdana,<br>arial, helvetica, sans-serif   | class="font2"       |
| font3           | Font option                | FONT: 8pt univers, verdana,<br>arial, helvetica, sans-serif   | class="font3"       |
| libraryselected | Selected Library           | BACKGROUND-COLOR:<br>#FF9933;   | COLOR: #ffffff      |
| librarypath     | Background color<br>option | BACKGROUND-COLOR: #f3f3f3   | class="librarypath" |
| lightwash       | Background color<br>option | BACKGROUND-COLOR: #f3f3f3   | class="lightwash"   |
| mediumwash      | Background color<br>option | BACKGROUND-COLOR: #f3f3f3   | class="mediumwash"  |
| menuitem        | Menu Items                 | COLOR: white;<br>FONT-FAMILY: univers, verdana,<br>arial, helvetica;<br>FONT-SIZE: 11px;<br>FONT-WEIGHT: bold                                 | class="menuitem"    |
| menulink        | Menu Link                  | FONT: 14pt univers, verdana,<br>arial, helvetica, sans-serif  | class="menulink"    |
| message         |                            | BACKGROUND-COLOR:<br>##EFEFEF;<br>COLOR: black;<br>FONT-FAMILY: univers, verdana,<br>arial, helvetica;<br>FONT-SIZE: 16px                     | class="message"     |
| mout            | Mouse Out                  | COLOR: darkblue   | class="mout"        |
| mover           | Mouse Over                 | COLOR: red  | class="mover"       |
| na              |                            | COLOR: #2a71ac;<br>FONT: bold 10pt univers,<br>verdana, arial, helvetica, sans-<br>serif  | class="na"          |
| nc1             |                            | BACKGROUND-COLOR: #c9e6ff   | class="nc1"         |
| nc2             |                            | BACKGROUND-COLOR: #f3f3f3   | class="nc2"         |
| nh              |                            | COLOR: #919191;<br>FONT: 15pt univers, verdana,<br>arial, helvetica, sans-serif   | class="nh"          |
| notselected     |                            | BACKGROUND-COLOR: #ffffff;<br>COLOR: #385273  | class="notselected" |
| selected        | Selected Option            | BACKGROUND-COLOR: #ffffff;<br>COLOR: #666699;<br>FONT-FAMILY: univers, verdana,<br>arial, helvetica;<br>FONT-SIZE: 11px;<br>FONT-WEIGHT: bold | class="selected"    |
| title           | Title                      | COLOR: #333366; FONT: 18pt<br>univers, verdana, arial,<br>helvetica, sans-serif   | class="title"       |
| toolbar         | Toolbar                    | BACKGROUND-COLOR:<br>#6BA8E6  | class="toolbar"     |
| upload          |                            | BACKGROUND-IMAGE:<br>url(/servlet/media/images/base/<br>toolback.gif);  | class="upload"      |

| Element/Class   | Description          | Attributes  | How to reference        |
|-----------------|----------------------|---|-------------------------|
|                 |                      | VERTICAL-ALIGN: top   |                         |
| white           |                      | COLOR: #ffffff  | class="white"           |
| wpadvice        | Large Instructions   | COLOR: #555555;<br>FONT-FAMILY: univers, verdana, helvetica; FONT-SIZE: 24px;<br>FONT-WEIGHT: bold  | class="wpadvice"        |
| wpcontentlist1  |                      | BORDER-BOTTOM: #6666CC;<br>BORDER-LEFT: #6666CC;<br>BORDER-RIGHT: #6666CC;<br>BORDER-TOP: ##EFEFEF  | class="wpcontentlist1"  |
| wpcontentlist2  |                      | BORDER-BOTTOM: ##EFEFEF;<br>BORDER-LEFT: ##EFEFEF;<br>BORDER-RIGHT: ##EFEFEF;<br>BORDER-TOP: ##EFEFEF   | class="wpcontentlist2"  |
| Wpdefaultcursor | Default cursor style | CURSOR: default   | class="wpdefaultcursor" |
| wpelemtoolbar   |                      | COLOR: #ffffff;<br>FONT-FAMILY: univers, verdana,arial,sans-serif;<br>FONT-SIZE: 8pt;<br>FONT-WEIGHT: bold  | class="wpelemtoolbar"   |
| wpoptions       | Options              | FONT-FAMILY: univers, verdana,arial,sans-serif;<br>FONT-SIZE: 8pt;<br>FONT-WEIGHT: normal;<br>TEXT-DECORATION: none   | class="wpoptions"       |
| wpselectedtitle | Selected title       | BACKGROUND-COLOR: #8CAAE7;<br>COLOR: #ffffff;<br>FONT-FAMILY: Arial, Helvetica, sans-serif;<br>FONT-SIZE: 9pt;<br>FONT-WEIGHT: bold;<br>TEXT-DECORATION: none | class="wpselectedtitle" |
| wptitle         |                      | BACKGROUND-COLOR: #e5eae;e;<br>FONT-FAMILY: univers, verdana,arial,sans-serif;<br>FONT-SIZE: 8pt;<br>FONT-WEIGHT: normal;<br>TEXT-DECORATION: none            | class="wptitle"         |
| wptoolbar       | Toolbar              | BACKGROUND-COLOR: #e5eae;e;<br>FONT-SIZE: 9pt;<br>FONT-WEIGHT: bold   | class="wptoolbar"       |
| wptreetop       | Background image     | BACKGROUND-COLOR: ##EFEFEF;<br>BACKGROUND-IMAGE: url(/servlet/media/templates/16/images/background.gif);<br>COLOR: white                                      | class="wptreetop"       |



**Figure 3-13: Style Sheet diagram for Infostore**



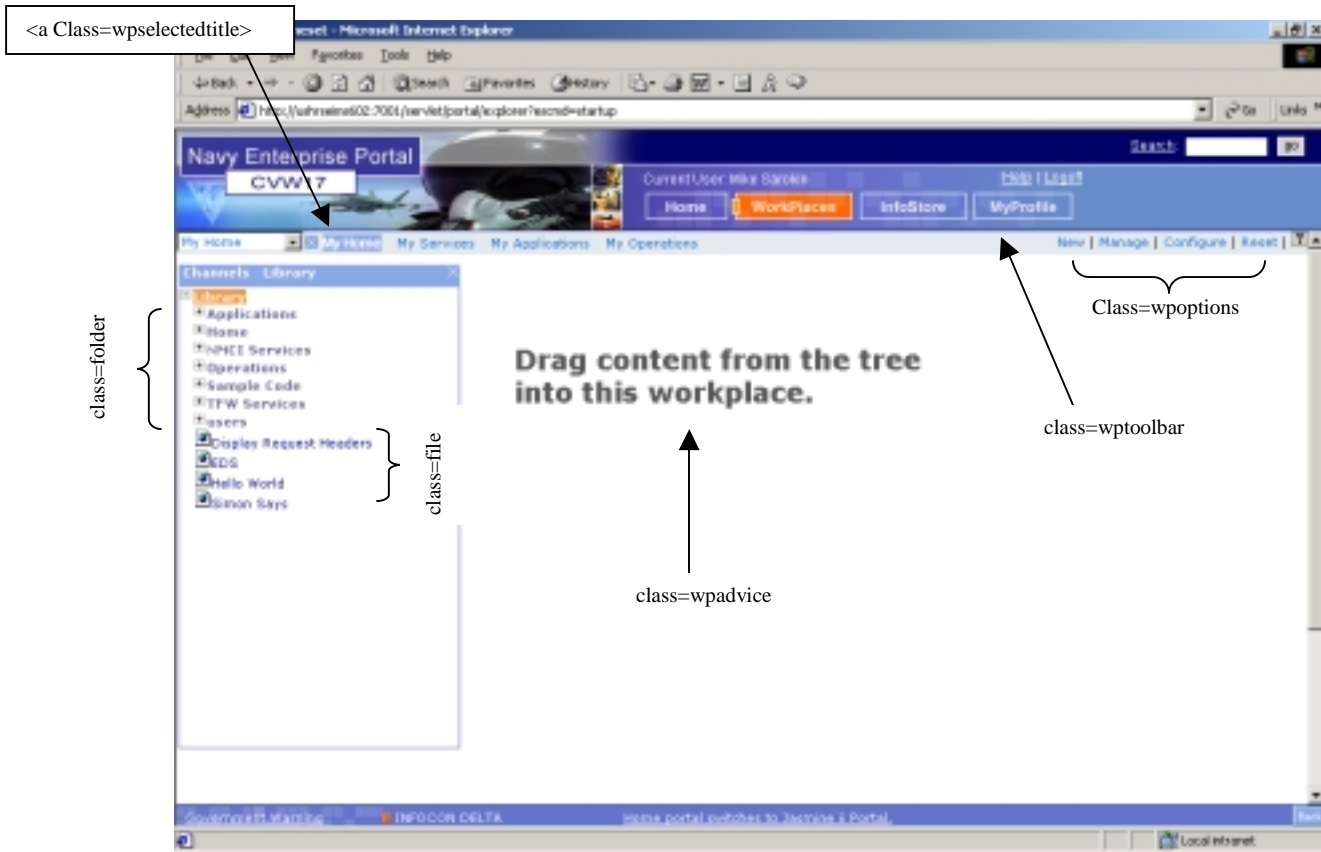


Figure 3-14: Style Sheet diagram for Workplaces

### 3.15.3 Portal Friendly Service Development

Service Developers *may* need to modify existing application code to have the service work properly in the TFW Portal. They may need to modify the application code to ensure the service works appropriately when accessed via Reverse Proxy from within the Portal Framework.

The CA Jasmine Portal reverse proxy feature handles the reconfiguration and rewriting of links to properly flow back through the portal infrastructure, but there may be potential reverse proxy issues with “absolute paths” versus “relative paths”. Some development may be necessary if the paths are generated dynamically or programmatically.

These service development recommendations can be considered by the Service Developers to ensure a portal friendly interface between their application and the TFW Portal.

#### 3.15.3.1 Images

HTML code sent by the service should contain Absolute Paths for the image SRC tag. If the HTML code contains Relative paths for the images then the BASE tag's HREF property must be set appropriately such that when the response is rendered in the client browser it can resolve the correct path for the images.

When the content from the Service is rendered through Portal Framework, relative paths for images without the proper BASE tag will result in unresolved image paths.

Usage:

*Relative Path:*

```
<HEAD>
  <BASE HREF="http://servername.navy.mil/service">
</HEAD>
<BODY>
  <IMG SRC="/images/test.jpg">
</BODY>
```

*Absolute Path:*

```
<IMG SRC="http://servername.navy.mil/service/images/test.jpg">
```

### **3.15.3.2 Links**

#### **3.15.3.2.1 Links for Integration Level 1:**

There is no restriction on Links in the response received from Integration Level 1 type services.

#### **3.15.3.2.2 Links for Integration Level 2 & 3:**

Links in the response received from Integration Level 2 & 3 type services will need to be re-configured, so that the request/response communication from these links goes through the Portal Infrastructure. To accomplish this, Links in the response should be only absolute and static paths (static paths here mean that the links are not generated using any client side JavaScript). These links are then re-configured by appending http path of the Portal as the pre-fix thus ensuring that all the communication to those links passes through the Portal Infrastructure.

If the Paths for these links are not absolute, the users request to access the link in that case will bypass the portal infrastructure and there will be no Portal to Repository Interface (PRI) data in the request header sent to the link. Additionally relative paths for links without the proper BASE tag will result in incorrect links and Service may not be accessible because of network boundary restrictions.

Example:

*Absolute Path:*

```
<A HREF="http://servername.navy.mil/service/ASP/test.asp">
```

Note: Both these scenarios are for Level 2 and Level 3 integration only and not for Level 1. PCS is mainly designed for Level 2 and Level 3 integration testing.

#### **3.15.3.2.3 Form Action**

The Action on the form needs to be always an absolute path, so that it can be reconfigured accordingly when the response is received from the service.

Because of technical limitations of the portal product (Reverse Proxy), Form Actions need to be Static and Absolute Paths. Thus on the service side any onSubmit JavaScript events relating to a Form should not modify it's Action. Additionally if the form Action is generated through JavaScript, it will bypass the TFW standard way of communication with applications, which dictates that all communication with services goes through the Portal Infrastructure.

Example:

```
<FORM NAME="TestForm" Action="http://servername.navy.mil/submit.asp"
METHOD="Post">
```

Amended Process Flow for PCS:

When the submit button is pressed on the PCS, Submit.ASP puts the PRI data in the Header and calls the URL which was entered in the "Enter URL" text box. The following process needs to be added upon receiving response from the service:

Receive the Request into the String and process as follows before it is written out to the Client Browser:

Re-configure Links and Actions as mentioned in Sections 2.2 and 3 above, so that any subsequent transactions relating to these links and actions are enforced to go through the PCS. This will ensure that all these links and actions receive the PRI data.

Different ways in JavaScript where you can specify ACTION dynamically.

If you are calling JavaScript method on an event which in turn calls the actual service then that JavaScript method should be directly available in the response and not as an included JavaScript file.

If links cannot be re-configured, it can cause a few issues:

PRI data will not be available in the Request Header.

Service may not be accessible because of network boundary restrictions.

Create the document in the following order:

- ÷! Issues – Mention issue & resolution then suggest
- ÷! (Do's) & Don'ts
- ÷! Actual design for PCS.

This version is to make sure that all the services called from within the service being tested receive the PRI data and that all the paths for images, actions, HREF's etc. are absolute paths and not relative.

### **Relative Paths for images and actions**

If the services use the HREF BASE tag with the root level directory of the service, they can then use relative paths for all images and actions in the forms (this will create a form action, which needs to be appended in some way with the ASP code to put PRI data in the Request Header).

### **Links needing Re-Configuring**

This is to load the PRI data in the request header before the request is submitted back to the service.

- ÷! <a href="http://" ></a>
- ÷! <a href="mailto:davey.jones@navy.mil">
- ÷! <a href="telnet://" ">
- ÷! <option value="http://" >
- ÷! <Form Action="testing.asp">

## **3.16 Portal Connector and Service Integration Process**

1. Portal Administrator will check if all the necessary information is submitted along with the Service Integration Request.
2. The Portal Administrator will check if the Service is registered in the Service Registry and is available in the Enterprise Service Repository.
3. The portal administrator will confirm that the Access Control List (ACL) is set in the Single Sign On product's database for the service under consideration.

4. The Portal Administrator will identify a portal developer and assigns the service integration work.
5. The portal developer will use the Portal Connector Template to create a new Portal Connector.
6. The Portal Administrator will publish the Portal Connector into the portal under the specified folder in the Portal connector taxonomy. The administrator will make the portal connector available to the Portal Developer ONLY. This will allow the portal developer to test the portal connector.
7. The portal developer will use the test plan submitted by the service developer to test the portal connector. Specific test goals will be to verify:
  - ÷! Application-level security model conforms to TFWeb guidelines e.g. ACL
    - Performance metrics.
      - ÷! Application responsiveness.
      - ÷! Multiple-application stress testing.
    - Conformance to TFWeb Look and Feel policies and guidelines.
8. The portal developer will submit the completed test plan and test cases to the portal administrator.
9. Portal Administrator will either accept or reject the service.
10. Portal Administrator will make the portal connector available to everybody unless otherwise specified. This does not mean that each and every user will be able to access the service. This only means that all the users can see the portal connector but the access to the service is controlled by the ACL set in the SSO database.
11. The portal administrator will send out a system message to announce the availability of the new service via the new portal connector.

### **3.17 How will User Browse / Access the Portal Connector**

The Portal Connector is published in the InforStore (Library) under the specified folder in the Portal Connector Taxonomy. The user will choose the workspace to place the new connector. The user will drag and drop the portal connector from the InfoStore (Library).

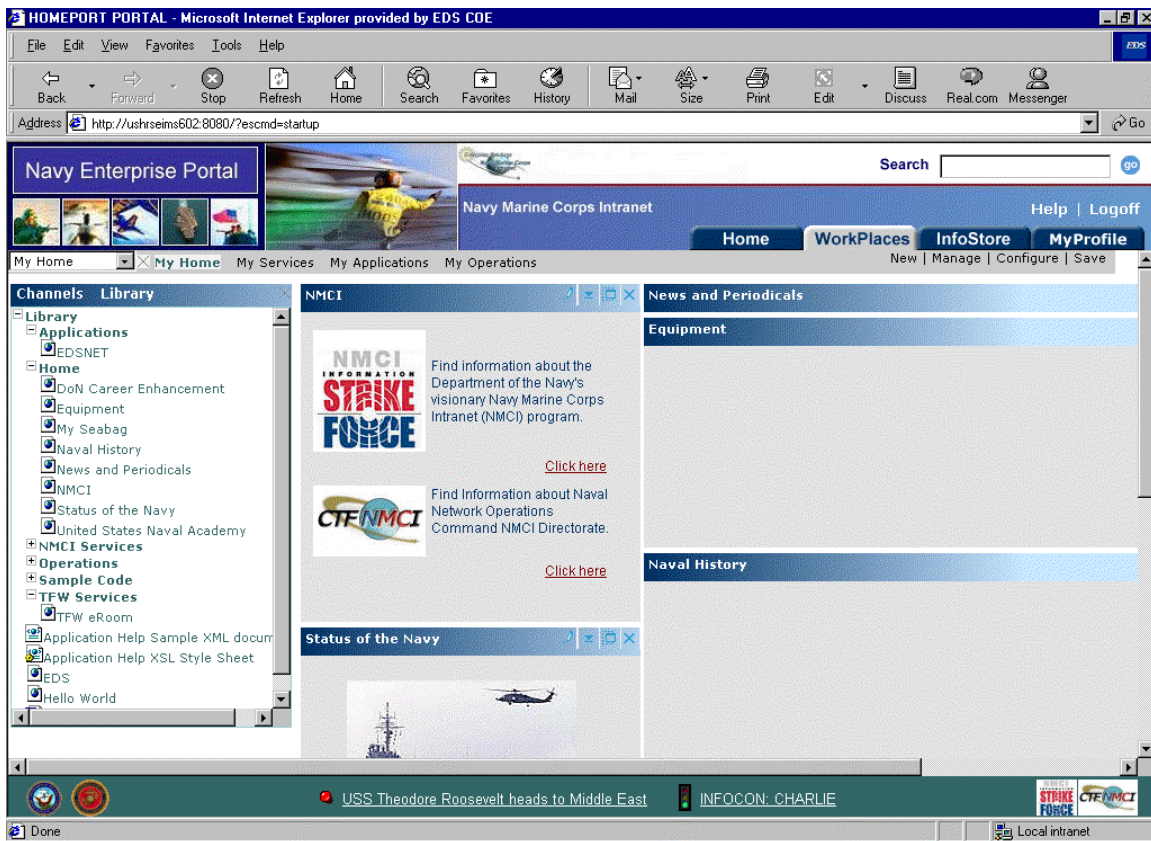


Figure 3-15: Portal Connector Library Screen Shot

At this time the Portal Connector will try to connect to the service and if the user has access to the service then the data will be displayed to the user. If the service is not available to the user (no permission) then a standard form will be displayed to the user stating the service owner contact information so that the user can contact the service owner and request access.

## 4 Web Service-Based Architecture

Web services is a term that defines a service-based infrastructure where application functionality can be identified, located, and accessed by users, or other applications, dynamically in near-real time. These services are implemented as software components, with no affiliation to a particular development platform or language. Information about these services, such as location, access methods and description, are housed within a global service metadata registry. Discovery and interaction with these services will be based on open, industry standards to ensure interoperability across applications and platforms. Numerous protocols either have already, or are in the process of being, defined to support web services, such as the Universal Description, Discover and Integration (UDDI) registry standard and the Simple Object Access Protocol (SOAP) XML messaging standard.

### 4.1 Key Concepts

The following are some key concepts of a web services-based architecture:

- ÷! The ability to browse through a common directory of available services
- ÷! The ability to access services using common, ubiquitous Internet communication protocols, such as HTTP
- ÷! The use of XML as a universal data format standard

### 4.2 Application and Information Services

The TFWeb portal implements web services through the application and information services layer, which consists of the service registry and service repository components highlighted in Figure 4.1.

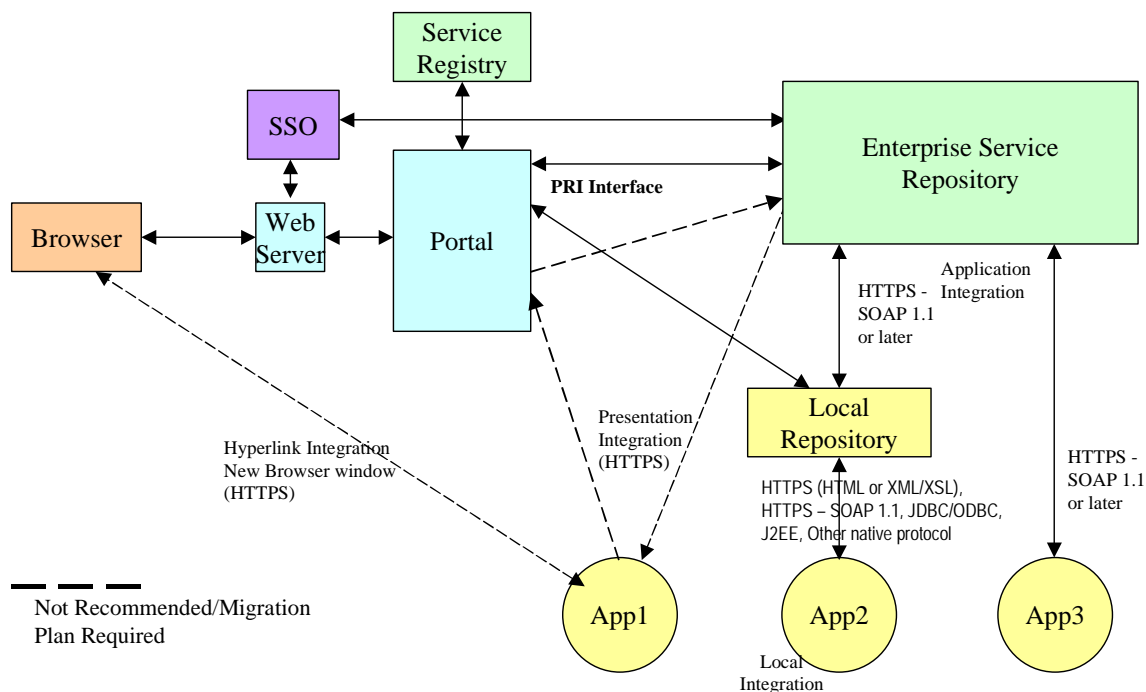


Figure 4-1: TFWeb Repository Architecture



### 4.3 Service Definition

A service module is a lightweight application connector that reveals some piece of application functionality and makes it available, in a web-enabled format, to end-users through the portal. A service module does not contain application business logic. Application business logic and data continue to reside within the application or its existing data store. Application owners are responsible for creating and maintaining the services.

At a minimum, a service module provides the following functionality:

- ÷! Decomposition of an incoming request
- ÷! Orchestration of information collection
- ÷! Authentication and authorization of users with external applications and data, as necessary
- ÷! Collection of information from external applications and data
- ÷! Composition of the associated response

Application owners are responsible for creating and maintaining the services. Services may be created using any capable application server platform or language, but they must conform to the requirements of the portal to repository interface described in Section **Error! Reference source not found.**

### 4.4 Service Repository

The service repository is a lightweight platform for the integration of applications with the enterprise portal. It is not a platform for hosting application business logic. Application business logic will continue to remain within the applications themselves.

The logic for communicating with the applications resides within the service modules. The service repository is the container for these service modules. The service repository is conceptually a single, logical construct, but physically it resides in multiple, distributed locations at multiple levels within the enterprise.

The Enterprise Service Repository consists of Microsoft Internet Information Server (IIS), and BEA WebLogic application servers. These application servers are the host platforms for Active Server Pages (ASP), Common Gateway Interface (CGI), and Java 2 Enterprise Edition (J2EE) service modules. The application developer may develop modules either repository server environment as best suited to the application requirements. All of the service modules must conform to TFWeb interface standards.

The function of each service module is to 1) receive user requests via the portal, 2) provide an interface to the underlying application logic and 3) provide the view logic for the responses back to the user via the portal.



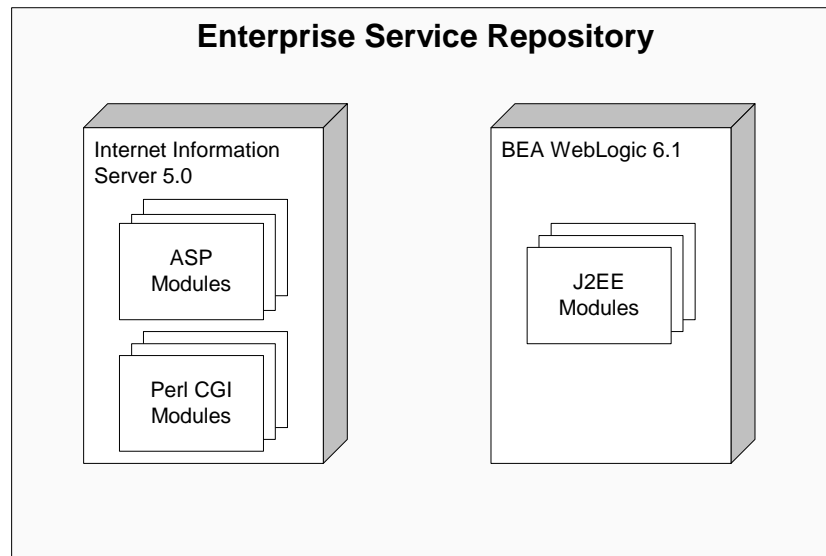


Figure 4-2: Enterprise Service Repository

#### 4.4.1 Enterprise-Level Service Repository

The Enterprise Service Repository provides an environment for application developers to host service modules near each portal instance. These service modules must conform to all TFWeb information assurance requirements, certification and accreditation requirements, portal and application interface standards, and naming standards. These systems also provide integration with the Single Sign-On (SSO) authentication.

The Enterprise Service Repositories are designed to be high-availability, and highly scalable.

#### 4.4.2 Claimant and Local -Level Service Repositories

The claimant and local-level repositories provides an environment for application developers to host service modules on non-TFWeb controlled systems. These service modules must conform to all TFWeb information assurance requirements, certification and accreditation requirements, portal interface standards, and naming standards.

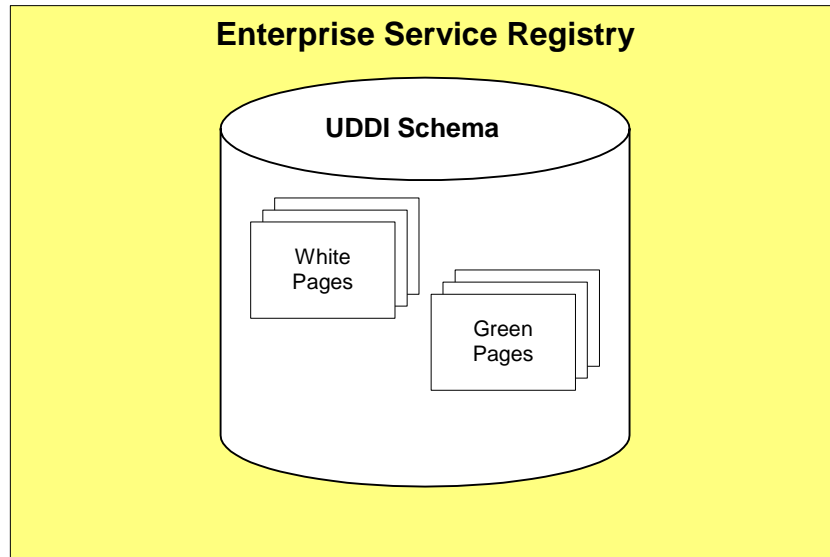
These systems do NOT 1) provide integration with TFWeb SSO authentication, 2) impose specific application interface requirements, or 3) require the use of specific application servers. Design, development, operation and maintenance of these types of repositories are not the responsibility of TFWeb.

Designers of these types of repositories should carefully consider the application scalability and availability requirements. The processing requirements and network bandwidth requirements justify a thorough analysis to determine the proper sizing of systems.

### 4.5 Service Registry

The service registry is a globally distributed registry of web services information. Each registry node consists of "white pages" and "green pages" for web services metadata. The UDDI data model has been selected as the basis for the TFWeb implementation. This model, as applied here, uses "white pages" entries to describe application owner information, while the "green pages" describe the services that can be accessed, including such things as the service name, description, and URI. The term "white pages" when, used in the UDDI context, should not be confused with the "Navy White Pages" directory.

As services are deployed to the service repository, they are registered in the service registry. This enables the portal to query the repository to determine the available services.



**Figure 4-3: Enterprise Service Registry**

Developers and portal end-users will have the capability to browse and search the list of services that reside in the registry. Developers and end-users will not have the capability to directly add, update or modify the contents of the service registry. The service registry administrator is responsible for adding, updating and maintaining contents of the service registry. Modifications to registry contents must be submitted as a request to the service registry administrator. The application owner supplies the metadata, and the administrator ensures that the application certification process is completed before the service is registered.

The metadata in the registry describe the application owners and their service modules. The relationship between white pages, green pages, and service modules is shown in the diagram below. Each application owner (white page) entry may have one or more registered services. Each service registration (green page) entry describes exactly one service module. Each application may comprise a number of service modules.

## Relationship between Registry Metadata and Repository Service Modules

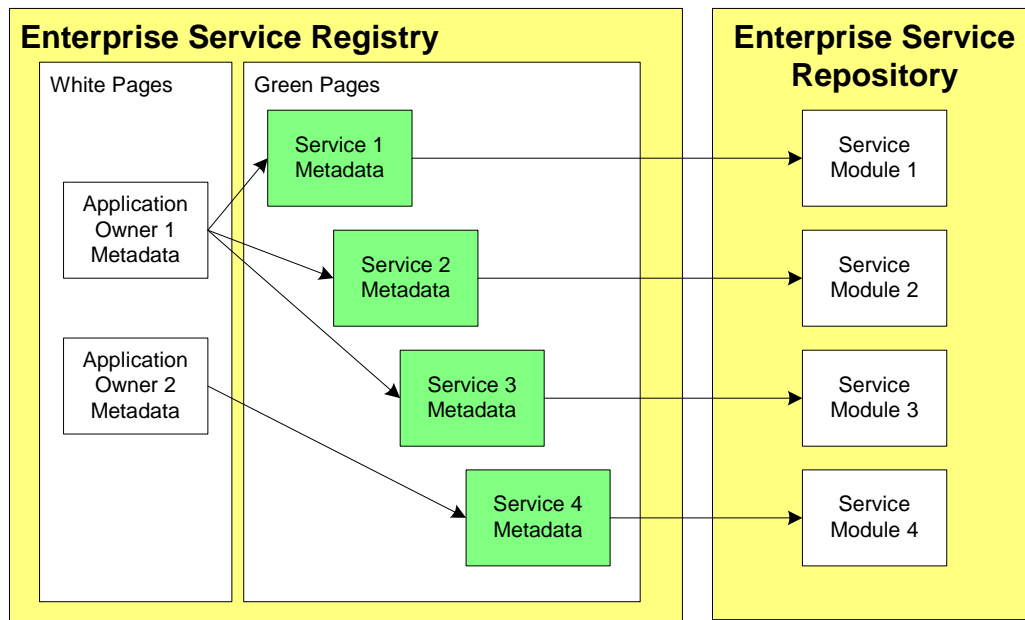


Figure 4-4: Relationship between Registry Metadata and Repository Service Modules

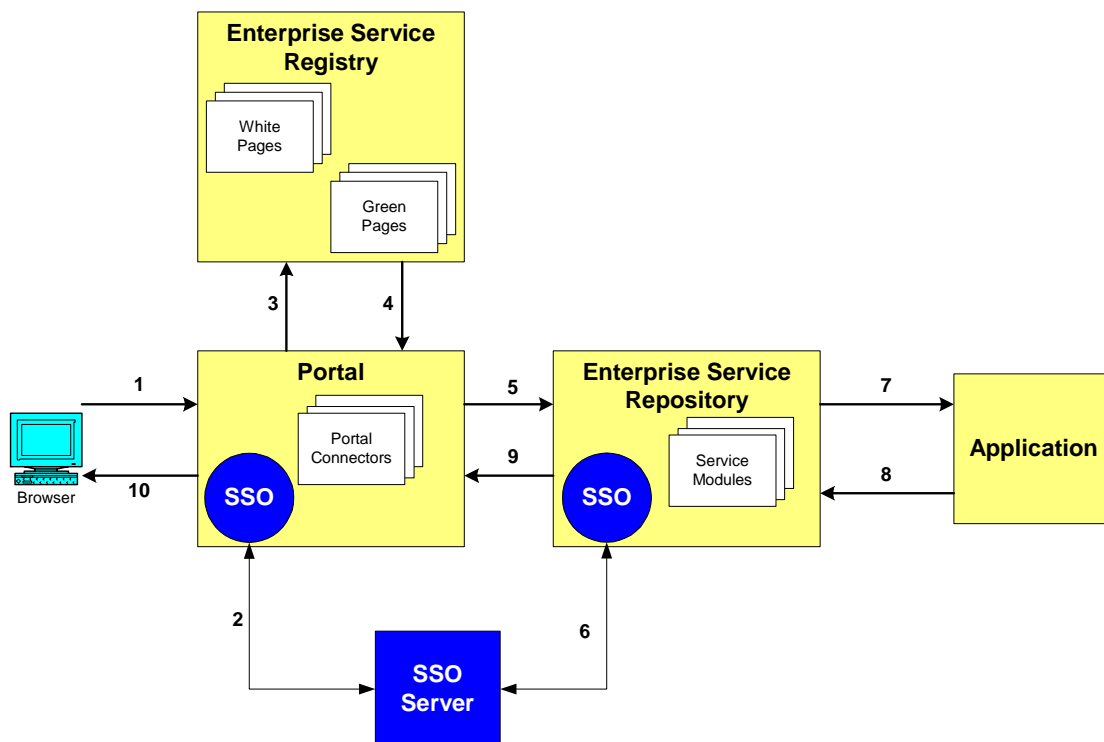


Figure 4-5: Application/Data Integration Process Flow Diagram

## 4.6 Portal to Repository Interface

All communication between the enterprise portal and the service repository occurs using the Portal to Repository Interface (PRI) specification. The PRI, as illustrated in Figure 4-1, is based on open, industry standards – specifically HTTPS and XML. The portal sends an HTTPS request to the URL that corresponds to the service being called. The request is an HTTPS “post” or “get”, with an XML message passed as an http header parameter. The XML message contains session channel context information for the request, such as the user identification and delivery channel, that the service may require in order to process the request.

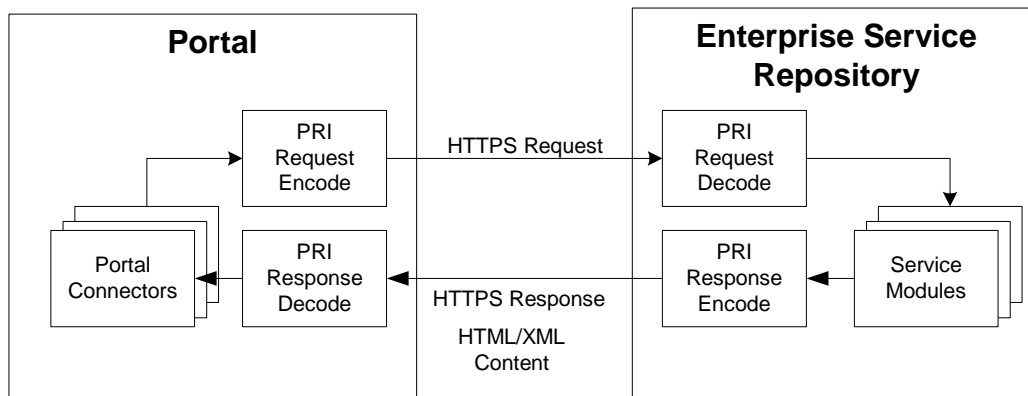


Figure 4-6: Portal to Repository Interface

After processing the request, the service sends a standard HTTPS response back to the portal. The content of this HTTPS response is either HTML, or XML and an XSL style sheet, which the portal will then render and display to the user. Included within the header of the HTTPS response is an XML message that includes some information and flags that the portal requires in order to render the response.

### 4.6.1 Transport Protocol

The PRI supports only Hypertext Transfer Protocol Secure (HTTPS) version 1.1 as the transport protocol. HTTPS implements secure, encrypted HTTP using the Secure Sockets Layer (SSL) standard. The SSL version 3.0 standard is required. All communication will be to the default HTTPS port 443.

The PRI interface supports both the HTTPS “post” and “get” methods. Please refer to the HTTP method definitions at <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9>.

#### 4.6.1.1 Support for Standard HTTP/HTTPS headers

The interface will support both read and write operations to the standard HTTP headers. Since the portal acts as an intermediary between the client browser and the repository, it will pass through the standard HTTP headers received from the client browser, in its request to the repository. In addition, any service repository (i.e. service module) modifications to the header, such as the setting of browser cookies, will also be passed through the portal to the client browser.

Please refer to W3C RFC 2616 (<http://ftp.isi.edu/in-notes/rfc2616.txt>) for a description of the standard HTTP headers.

### 4.6.2 HTTPS Request

The enterprise portal will be the source system initiating the PRI HTTPS request. The request will be triggered when an enterprise portal user selects the portal connector in his/her portal

workspace, resulting in an HTTPS “post” or “get” generated from the portal to the URL/URI that corresponds with the appropriate service module within the repository.

#### 4.6.2.1 Request Data Definition

Table 4-1 defines the contents of the portal's HTTPS request to the service module. This includes the PRIDataRequest message, which is an XML message generated by the enterprise portal and passed as an HTTP header variable in the initial HTTP request. This key value for this HTTP header variable is 'PRIDataRequest'. Please see Appendix B for the XML schema associated with the PRIDataRequest message.

**Table 4-7: PRI Request Data Definition**

| Data Element Name  | Size / Format                 | Description  | Notes  |
|--|-------------------------------|--|--|
| Standard information to be sent as part of the HTTPS request       |                               |  |  |
| Standard HTTP Request Headers                                      | See section 4.6.1.1.          | Standard HTTP headers that the portal received from the client browser.    | HTTP headers are passed in the request from the source to target system reflect the header information received from the client browser via web infrastructure.  |
| PRIDataRequest data elements sent as a variable in the HTTP header |                               |  |  |
| UserID   | 200 characters / Alphanumeric | The portal user's identification based on the Navy flat name space schema. | The portal framework must determine the user ID from either the client browser or the directory service.   |
| RoleAssignments  | Array of Alphanumeric Strings | The user's role assignments.   | The portal framework must determine the user role assignments from the directory service.  |
| PortalLocation   | 80 characters / Alphanumeric  | The location of the portal instance.                                       | Either “ashore” or “afloat”. For the Pilot, the portal is not required to dynamically determine this value. It may be manually configured within the portal connector template instance.                           |
| Client   | 80 characters / Alphanumeric  | The content delivery channel to the client.                                | For the Pilot, the only supported delivery channel will be “browser”. The portal is not required to dynamically determine this value. It may be manually configured within the portal connector template instance. |

| Data Element Name | Size / Format                | Description   | Notes   |
|-------------------|------------------------------|---|---|
| CheckBandwidth    | 10 characters / Alphanumeric | A flag to inform the service module that communication bandwidth restrictions may exist for this request. | This value will be either “true” if the service module is required to verify bandwidth availability, or “false”. The portal is not required to dynamically determine this value. It may be manually configured within the portal connector template instance. |
| SessionID         | Character string format GUID | A globally unique session identifier for the portal user's browser session.                               | A 32-character Globally Unique Identifier (GUID) in the format “nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn”. The portal generates this value dynamically.   |
| PortalName        | 40 characters / Alphanumeric | The fully qualified DNS name of the portal that originated the HTTPS request.                             | For the Pilot, and also in the future, it is most likely that there will be only one logical portal i.e. one portal name.   |

#### 4.6.2.2 Timeouts

The PRI interface currently does not provide the capability to dynamically set the timeout value to wait for a response from the service repository for each request / reply transaction. The timeout value is currently a static value, configurable by the portal administrator.

##### 4.6.2.2.1 HTTPS Response

The target system for the PRI HTTPS Request from the enterprise portal will be the service repository, and specifically the service module that resides within the service repository. The service repository is a logical concept that may exist at many levels – enterprise, claimant and local. Claimant and local repositories are the responsibility of the specific claimant or local application owner. Claimant and local repositories may be implemented in any manner, but must be compliant with the portal to repository interface specification in order to integrate with the TFWeb portal.

#### 4.6.2.3 Response Data Definition

Table 4-2 provides the data definition of the service module response to the portal.

**Table 4-8: PRI Response Data Definition**

| Data Element Name   | Size / Format       | Description  | Notes   |
|---|---------------------|--|---|
| Standard information to be sent as part of the HTTPS request. |                     |  |   |
| Standard HTTP/HTTPS Response Headers                          | See Section.4.6.1.1 | Standard HTTP response headers that the portal received from the client browser. | HTTP/HTTPS headers are passed in the request from the source to target system reflect the header information received from the client browser via web infrastructure. |

| Data Element Name   | Size / Format                 | Description   | Notes   |
|---|-------------------------------|---|---|
| Reply coming back from the service                                  |                               |   |   |
| Content   | XML + XSL (preferred) or HTML | The content returned from the service to be rendered by the portal and displayed in the client browser.   | The service module must respond with portal compliant HTML. Please see the TFWeb Portal Service Architecture Design Document for more details concerning portal compliant HTML requirements.                                      |
| PRIDataResponse data elements sent as a variable in the HTTP header |                               |   |   |
| IntegrationType   | Numeric                       | The TFWeb-defined level of integration being provided by the service module.  | Either "1" for Hyperlink Integration, "2" for Presentation Integration or "3" for Application/Data Integration.   |
| ReconfigureLinkFlag   | Boolean                       | A flag returned by the service to specify if hyperlinks returned in the service content should be reconfigured by the portal reverse proxy.   | Either "true" if hyperlinks shall be reconfigured, or "false" if hyperlinks shall not be reconfigured. Please see the TFWeb Portal Service Architecture Design Document for more information concerning reconfiguration of links. |
| Timeout   | Numeric (Integer)             | A numeric value returned by the service module to specify, in seconds, the default request timeout value for subsequent portal to repository requests made by that specific portal connector. | A numeric integer value, greater than zero. This is a future capability that will not be supported in the Pilot.  |

#### 4.6.2.4 Content Requirements

The HTML that is returned by the service module, or as a result of the combination of the returned XML data and XSL stylesheet, must be compliant with the W3C HTML 4.0 specification. The HTML must also be portal compliant as defined in the TFWeb Portal Service Architecture Design Document.

##### 4.6.2.4.1 Error Handling

The service modules must generate standard HTTP error return codes in the case of an exceptional or abnormal condition. For example, error code 401.3 will be returned in the case



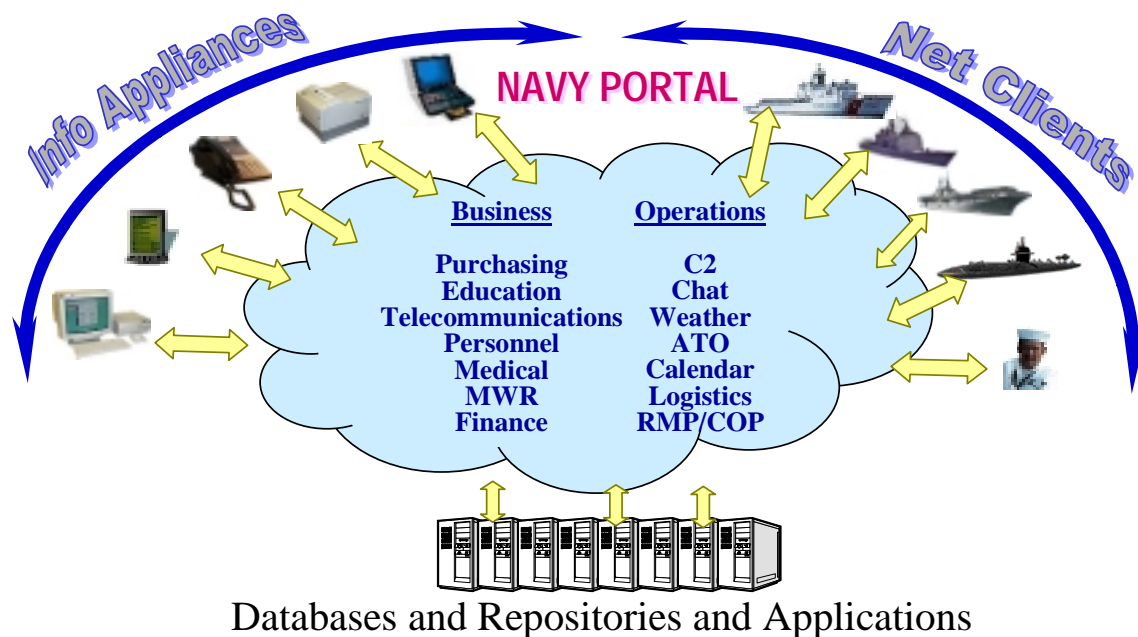
that the user does not have authorization to access the requested service module. In the case of service modules located within the Enterprise Service Repository, authorization to access services is controlled through the permissions management application. The permissions management application provides a proxy plug-in for the Enterprise Service Repository that protects the URI associated with the service module, returning the service module content if the user has access, or returning an HTTP 401.3 error code if the user does not have access. The portal connector template makes use of this behavior to respond appropriately. In the case that a user does not have access to a specific service module, the portal connector will display an appropriate error message and provide a standardized form to request access to the service.

Claimant/Local Service Repositories will not have the permissions management proxy plug-in, but are still required to implement the same behavior as the Enterprise Service Repository. Please see the TFWeb Portal Architecture Design Document and the TFWeb Directory Services/Permissions Management Architecture Design Document for more information regarding the service module access authorization requirements.

Please refer to <http://www.w3.org/Protocols/rfc2616> for details on standard HTTP error codes.

## 5 Enterprise Portal Taxonomy

Navy web enablement is the implementation of interoperable web technologies across the Naval infrastructure allowing subscribers and publishers (users and providers) of content to pull or push services as required to perform operational or business transactions. A Navy web transaction is the execution of a web-service. The service centric access for the Navy is depicted in Figure 5-1.



**Figure 5-1: Service-Centric Access**

This section will discuss guidance and structure for the TFWEB Portal taxonomy. This will provide developers with the appropriate background to plan their application migration efforts. This will section will be further refined by the Navy Portal Integration Office (PIO) as more is understood about the available services.

At the highest level, the taxonomy represents the basic set of categories for Navy information sources. The information involved may be core to the function being performed, or to some other functional area. The TFWEB Portal System facilitates the sharing of information between commands and across functional areas. The Department of Navy Chief Information Officer (DoN CIO) has identified broad information content categories that reside within the enterprise. Table 5-1 identifies the initial, high-level set of categories.

**Table 5-2: Initial TFWeb Portal Taxonomy**

| Functional/Resource Area | Program/Resource Sponsor |
|--------------------------|--------------------------|
| Acquisition              | SECNAV RDA/ MARCORSYSCOM |
| Finance                  | SECNAV FM&C/ HQMC P&R    |
| Civilian Personnel       | SECNAV CP/ HQMC AR       |

|  |                        |
|--|------------------------|
| Administration                           | OPNAV N09B/ HQMC AR    |
| Manpower and Personnel                   | OPNAV N1/ HQMC MR&A    |
| Intelligence and Cryptology              | OPNAV N2/ HQMC I       |
| Logistics                                | OPNAV N4/ HQMC I&L     |
| Readiness                                | OPNAV N4/ HQMC PP&O    |
| Command, Control and Communications      | OPNAV N6/ HQMC C4      |
| Information Warfare                      | OPNAV N6/ HQMC PP&O    |
| Allies                                   | OPNAV N6/ HQMC PP&O    |
| Modeling and Simulation                  | OPNAV N6/ MCSC SE&I    |
| Weapons                                  | OPNAV N7/ MARCORSYSCOM |
| Training                                 | OPNAV N7/ TECOM        |
| Resources, Requirements, and Assessments | OPNAV N8/ HQMC P&R     |
| Scientific and Technical                 | OPNAV N091/MCCDC       |
| Test and Evaluation                      | OPNAV N091/ MCOTEA     |
| Medical                                  | OPNAV N093             |
| Naval Reserve                            | OPNAV N095             |
| Meteorology, Oceanography, MC&G          | OPNAV N096             |
| Religious Ministries                     | OPNAV N097             |
| Naval Nuclear Propulsion                 | OPNAV NOON             |

## 6 Security

### 6.1 Security Definitions

A few terms are used in this section that are, at times, confusing or are used in a more generic manner in a broader context. These definitions refer specifically to the TFWeb Single Sign-On (SSO) environment.

- **Authentication.** The process of verifying that a user is, in fact, the user they claim to be. Authentication in the TFWeb Pilot will be based on userID and password credentials. Future implementations will include the use of DoD PKI X.509 identity certificates for authentication. The process of authentication does not grant the user with any particular rights.
- **Authorization.** The process of determining what rights are granted to a user based on the user's authenticated identity.

### 6.2 TFWeb Security Architecture

RSA Security's ClearTrust SecureControl (CTSC) is the product that will provide the SSO solution for the TFWeb architecture. RSA's ClearTrust product:

- Acts as a central point of authentication for enterprise Portal users,
- Provides a framework for authenticating users which will enable a Navy-wide web SSO implementation by Application owners,
- Authorize use of Portal services and Applications based on dynamic security policies configurable by Application owners,
- Utilize enterprise Active Directory architecture by replicating users, user attributes, and security groups to establish Portal security policies.

The ClearTrust SecureControl solution provides a single, unified mechanism and interface for controlling access and security across platforms, applications, and Web servers. More specifically, ClearTrust SecureControl's authentication and authorization solution provides SSO centralized management across platforms and vendors, delegated user management, rules-based access control, and support for multiple forms of authentication.

The SSO solution will provide a framework to allow Navy Portal users to move seamlessly across the Portal web servers and Applications without having to re-authenticate each time they click a new link. Their authentication information is passed on to other SecureControl components via an encrypted session cookie. SSO depends on the storage of authentication information in encrypted session cookies. At runtime the plug-in communicates with the Key Server, which maintains a list of randomly generated session keys that the plug-ins use to encrypt and decrypt session cookies. These keys are changed regularly, but at any given time, all of the plug-ins use the same list of keys. Therefore, a plug-in can always decrypt a cookie generated by itself or any other Web Server Plug-in in the system, and can authenticate using stored information rather than re-prompting the user for credentials.

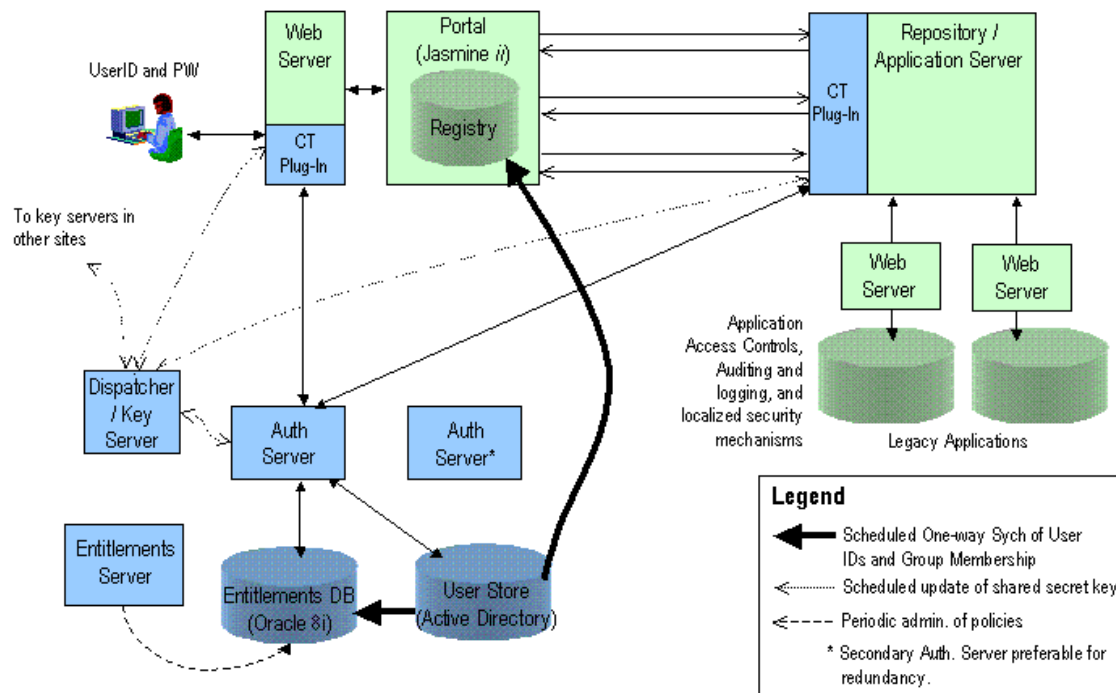


Figure 6-1: TFWeb Security Architecture

### 6.2.1 Role of the Directory in SSO

The enterprise-wide TFWeb Active Directory architecture is the basis for all Navy Portal SSO operations. The TFWeb directory architecture will be replicated/synchronized across the Navy enterprise, and will be collocated with the Portal architecture components. This directory architecture will also extend to include Navy afloat ship platforms.

The primary function of the TFWeb Active Directory architecture is to provide a centralized location for the authentication and authorization of all Navy and USMC users to the TFWeb Portal architecture.

In order to be the centralized source for all user authentication, the TFWeb directory will establish global unique user identifiers for all registered Navy and USMC TFWeb users, both afloat and ashore. These user identities will be based on the flat SMTP name space being fielded by the NMCI (e.g., [joesph.user@navy.mil](mailto:joesph.user@navy.mil), [joesph.q.user@navy.mil](mailto:joesph.q.user@navy.mil), [jane.user@usmc.mil](mailto:jane.user@usmc.mil), [jane.j.user12@usmc.mil](mailto:jane.j.user12@usmc.mil)). Every user will be assigned a new unique ID in the flat name space that will remain theirs, regardless of their location within the Navy or USMC organizations.

Authentication of users to the TFWeb Portal and its Applications will be based on the flat name space userID. The ClearTrust SSO product will perform authentication of a user's identity directly against the TFWeb Active Directory.

The TFWeb Active Directory will also be the centralized source for authorizing user access to TFWeb Applications. Active Directory groups will be defined which determine a user's ability (or inability) to use a particular Portal Application. The ClearTrust SSO product will utilize these groups in allowing or denying users access to the Portal and its Applications.

The TFWeb directory service is the initial implementation of an enterprise-wide directory to support Navy and USMC applications, such as the WEN. The TFWeb directory is the basis for, and will eventually evolve to become, the larger Naval Global Directory Service (NGDS). The NGDS will provide enterprise-wide services such as TFWeb authentication, location of Navy and

USMC personnel (Navy/Marine Corps White Pages), and convergence of NMCI and IT-21 directory services into a logical global directory

### 6.2.2 SSO Architecture Components

The ClearTrust SecureControl SSO service is comprised of the following components:

- **The Entitlements Server.** The Entitlements Server is the central server for the *administrative* functionality of the ClearTrust SecureControl system. All security policy changes to the Entitlements Database are made via the Entitlements Server. Whenever you update your user, resource or security policy information, you can also clear the cached data on the Authorization Servers via the Entitlements Server, so that the updates take effect immediately.
- **The Authorization Servers.** The Authorization Servers perform the authorization checks at runtime. When a user tries to access a resource, the Authorization Server essentially asks two questions: “does these user credentials prove that this user is valid?” and “does this user have access to this resource?”
- The Authorization Server will authenticate a user’s identity against the external Active Directory User Store. The Authorization Server reads policy information directly from the Entitlements Database. However, all approved requests are cached in the Authorization Servers, thereby improving the runtime performance. One primary and one back-up Authorization Server is installed for a standard set-up of ClearTrust SecureControl.
- **The Authorization Dispatcher/Key Server.** This component provides two functions. The Dispatcher keeps track of all the Authorization Servers that are available. When an access request comes in from a Web Server Plug-in, the Dispatcher routes the request to the next available (or geographically closest) Authorization Server.
- The Key Server is a separate process that generates new token encryption keys (or session keys) periodically. The Web Server Plug-ins query the Key Server to get the latest key. When users authenticate to the SecureControl system, they are issued encrypted cookies that identify them in future interactions with the system. These cookies are actually small, encrypted pieces of data that contain session information. The Web Server Plug-ins are the only components that can generate these cookies. In order that any SecureControl Plug-in can decrypt any session cookie, even those generated by other Plug-ins, all must use the same encryption key. For security reasons, this key is changed frequently.
- **The Entitlements Database.** The Entitlements Database stores web server, application, and entitlement information. The purpose of the Entitlements Database is to relate a user’s identity with the web resources they are or are not allowed to access. In the Entitlements Database, a user may be granted access to specific resources or explicitly denied access to resources. This will be done by allowing access to resources based on the membership of Groups defined in the Active Directory user store.
- **Web Server Plug-In.** The ClearTrust Web Server Plug-In implements SSO by creating and interpreting HTTP cookies that contain a user’s session token. The user’s browser passes the cookie to the Web Server Plug-In as proof of the user’s authenticated identity. When a SecureControl Web Server Plug-In receives a cookie, it extracts the token and then extracts the identity of the user and user’s authentication mechanism from the token. If the user is authorized to use the requested resource, then a new token is generated with an updated timestamp and returned to the user’s browser.
- **The Active Directory User Store.** The Active Directory user store is actually just one site of the TFWeb enterprise directory architecture. This Active Directory site contains all of the users, user credentials, and groups that will be used to authenticate a user’s identity, and manage user’s access rights for the TFWeb Portal and Applications. The Active Directory user store is one-way replicated (on a scheduled basis) into the

Entitlements Database to establish the user identities and group memberships upon which the ClearTrust security policies are based. All changes to the access list for a particular Application will occur in the TFWeb Active Directory architecture in the form of the change in membership of an Active Directory security group.

### 6.2.3 How TFWeb SSO Works

This is the process that takes place when a user attempts to access web resources that are protected by the ClearTrust SSO product.

- A user requests a web resource on a ClearTrust protected web server.
- The web server plug-in checks the Authorization Server cache (for enhanced performance), or queries the Entitlements Database (RDBMS) to see if the requested URL is a protected resource, and what credentials are required.
- If the resource is not protected, the user can access it immediately.
- If authentication is required, the user is prompted for a userID/password, or some other credentials. During the TFWeb Pilot, userID and password will be the primary means of access to the TFWeb Portal system. In the future, additional credentials will be supported, particularly DoD PKI certificates.
- The user provides the credentials.
- The Authorization Server authenticates the user against the TFWeb Active Directory server using the credentials provided.
- The Authorization Server determines the user's rights to the requested URL resource by either checking its cache, or querying the Entitlements DB for the user's security policies.
- Access permission information - either "allow or deny" is cached in the Authorization Server and sent to the Web server plug-in.
- The web server Plug-in allows or denies the user request based on the user's permissions. When the web server Plug-in allows the user request, a session ID cookie is created and sent to the user's browser.
- When the user requests another protected Application from the Portal, the session cookie is passed along to the Web Server Plug-in with all of the user's identity information.
- The Web Server Plug-in gets the secret key from the Dispatcher/Key Server to decrypt the session cookie. Once the user's information is decrypted, the Web Server Plug-in can then authenticate for the new resource using the information in the cookie. The user is not re-prompted for their username and password.

When the Plug-in attempts to decrypt the session ID using the symmetric key, if the decryption fails, the user is forced to re-authenticate. By default the cookie is stored in memory and not written to the user's hard disk. The cookie is encrypted using 3DES.

## 6.3 TFWeb Security Model - Allow then Deny

The TFWeb architecture will follow an 'Allow then Deny' approach to security for the Portal and backend Applications, i.e., all of the Application connectors to the Portal are visible to (nearly) all of the users. A minimum number of Portal workgroups will be created to define which Portal connectors certain user groups will be able to see. There is likely to at least be a Default group, a Navy group and a USMC group. Each group will define a large number of connectors that the users will be able to see. Based on Department of Navy (DoN) policy, there may be a very few exceptions (in case of very high security applications), where the Portal connectors will only be visible to a few authorized users. Thus, majority of the portal connectors will be visible to all the workgroups in the Portal.



Portal users can arrange the Portal connectors on their workspace of choice by dragging and dropping them into their display. At this point in time, the Portal connector will try to connect to the underlying Application by sending an HTTPS request to the Applications URI on the Repository web server. The ClearTrust plug-in on the Repository web server will then either allow or deny access based on the user's rights to the Application as defined in the Entitlements Database. Thus, in some cases, even though the Portal connector is visible to the end user, the actual contents of the Application may not be available. The user will need to request access to the Application from its owner.

In case of an Application's Portal connector, access control is achieved through the security policies set in the ClearTrust Entitlements Database. The ClearTrust plug-in installed on the Repository web server will challenge any requests for Applications and allow or deny access based on the security policies set for the Application. It is the Application owner's responsibility to provide the necessary information to the directory and SSO administrators to set the Access Control List (ACL) policy for the Application. This information will consist of the Application's name, and the definition of all users that require access to the Application. This definition of users may be based on a role attribute (assuming it is an attribute established in the TFWeb Active Directory) or by user name. Once the authorized users are defined, the directory administrator will create an Active Directory group that, in conjunction with the ClearTrust product, will allow access to the Application.

## **6.4 Application Access Control Mechanisms**

### **6.4.1 ClearTrust Application Access Control Mechanisms**

ClearTrust provides a very important component of the overall enterprise TFWeb system security. The Web Enabled Navy architecture endeavors to provide Navy and USMC users with a single point of access to all web enabled applications. ClearTrust provides the top-level, enterprise security mechanisms to provide an integrated TFWeb Naval web application environment, even though these web Applications will continue to be developed and maintained by multiple Naval organizations.

The SSO mechanism is important to provide an integrated view of the TFWeb Applications. The ClearTrust product and TFWeb Active Directory architecture will act as the central point of authentication for all enterprise Portal users. The goal of the Navy Portal is that this initial authentication of the Portal user should be the only authentication necessary for use of all the TFWeb Applications. This is because the ClearTrust SSO mechanism can pass authenticated user identities to the Portal, and all backend web Applications, for their use in assigning rights and roles within the Applications.

ClearTrust SSO will provide the first point of authorization for use of the Portal and all portal Applications. ClearTrust provides the means to approve or deny access to portal Applications (as represented by the web URLs pointing to those Applications), but cannot provide users with detailed rights to the underlying Application. For example, ClearTrust can authorize an authenticated user to access an Application that utilizes an internal database, but cannot specifically grant or deny the user the right to read, write, or administer the database itself. Local security for the Application is still the responsibility of the Application developer.

Lastly, the ClearTrust SSO shall perform auditing and logging of failed and successful attempts to access the Portal and portal Applications.

#### **6.4.1.1 User Passwords**

Each user in the TFWeb Active Directory will have a password. ClearTrust support other forms of authentication, such as X.509 Certificate authentication, but the Pilot implementation will be concentrating on the userID and password combination. Whenever a password is passed over the network or stored on disk there is a risk that password may be stolen. In order to reduce the likelihood of password theft, TFWeb is using multiple mechanisms:

- Passwords are stored in Active Directory using a Kerberos encrypted password store, making the original password unrecoverable.
- Passwords are never stored, in clear text or encrypted, anywhere but the Active Directory user store.
- Passwords are always sent over SSL encrypted network connections from the user's browser, to the Web Server Plug-In, to the Authorization Server.
- Authentication of a user's ID/password credentials is actually performed by the ClearTrust Authorization Server, which will be installed resident on the same physical hardware as the Active Directory user store.

### 6.4.2 Legacy Application Access Control Mechanisms

As stated above, ClearTrust will provide enterprise-level security, including SSO authentication and authorization to access the URL that defines the 'home page' for a portal Application. However, it is the responsibility of the Application owner/developer to provide local security, at the Application level. This security includes determining the process by which users are authenticated and authorized to the legacy Application.

To participate in the overall SSO solution for TFW, Application developers must:

- Determine the means by which users will be authenticated to the legacy Application through the Portal (see section 7.4.3)
- Map TFW userIDs to the Application's internal user rights database (for legacy Applications) or develop a user rights database using the TFW userIDs (for new Application development)
- Provide TFWeb SSO administrators with a list of users authorized to use the Application.
- Authorize user's rights to internal Application processes or databases based on the authenticated userID provided by ClearTrust.

In particular, the Application must assign all rights that are necessary internal to the Application. Examples of these user's rights include: allowing users to modify data, providing access to portions of internal databases, and administering other user's rights.

Application developers must also continue to provide sound local security for their web servers and Applications. This local security may include mechanisms such as applying all appropriate patches for the operating system and web server software, providing full support for encrypted web communications (i.e., HTTPS) using DoD PKI server certificates, and protecting the web server and Application from common hacker attacks such as IP spoofing and denial of service. The Application must also provide local auditing and logging of access attempts and invocations of user rights.

Refer to Appendix A: DoD Authority References for a complete list of Navy and DoD security policies to be followed.

Security certification and accreditation of the Application as a stand-alone service (following all pertinent Navy guidelines and policies) remains a responsibility of the Application developer. Applications that are not accredited will not be allowed to take part in the TFWeb architecture.

### 6.4.3 Legacy Application Authentication Options

To implement SSO, the ClearTrust product relies on encrypted cookies to securely pass authenticated user IDs between web applications. ClearTrust also provides a means to pass authenticated user IDs through the HTTP request header fields (see Section 7.5.2). However, TFWeb recommends that Application developers do not use this data as a means to automatically authenticate users to their Application. Because this data may possibly be spoofed, it should not be trusted as an authentication means, but may be used to provision "familiarity" services, such as a personalized splash page, etc.

This leaves Application owners with a few options to authenticate users to the Application, as described below.

- **Re-Authenticate Users:** During the TFWeb Pilot timeframe, the Application owner may choose to re-authenticate users accessing their Application through the Portal. As this solution does not lead to the goal of end-to-end SSO for the Portal, the Application owner will be asked to migrate to the fully integrated SSO solution once it is determined.

Application developers may still be required to change the interface through which users authenticate, even if the backend authentication process remains the same. Two interfaces are acceptable for integration into the Portal:

**HTML Form:** The Application should present the user with an HTML form requesting credentials. This form should be designed in-line with the rest of the Application's interface, and should not be a new pop-up browser window.

**Dialog Boxes:** This is the less preferred, but acceptable authentication interface. The Application may present the user with a system-generated dialog box requesting credentials. However, this dialog box **MUST** uniquely identify the Application that is asking for credentials to differentiate it from any other dialog boxes in use.

- **Implement ClearTrust Web Server Plug-In:** Application owners may choose to implement a ClearTrust Web Server Plug-In on the legacy Application's web server. Using the ClearTrust Web Server Plug-In will tightly integrate the Application with the Portal SSO solution, which is the long-term goal. There are no licensing issues related to using the ClearTrust plug-in, but there are some technical limitations. See Section 7.5.3 for a discussion of the use of the Web Server Plug-in.

Regardless of the authentication solution chosen for the Application, Application owners are still required to provide a list of all users authorized to access the Application (as stated in Section 7.5.1) to allow proper operation of the enterprise SSO solution.

It must also be noted that any change to the Application's authentication process should result in a re-evaluation of the Application's security posture by the System Security Approval Authority (SSAA).

## 6.5 Application Interactions with SSO/AD

### 6.5.1 Authorizing User's Access to the Application

Application owners will maintain full, positive control for access to their web Applications through the Portal.

As has been stated above, authorized access to resources will be based on the membership of security Groups defined in the Active Directory user store. During the TFWeb Pilot, the Application owner must provide the directory and SSO administrators a list of users who should be authorized to access the Application. Once the authorized users are defined, the directory administrator will create an Active Directory group that, in conjunction with the ClearTrust product, will allow access to the Application.

This list of users shall be provided in either an LDAP Data Interchange Format (LDIF) file or a Comma Separated Variable (CSV) file that contains the following data about the users:

- First Name,
- Last Name,
- Middle Initial,
- Rank (if military),
- E-mail Address,

- Navy Portal Flat Name Space UserID,
- Application Specific UserID.

In the future, a web application will provide Application owners with the capability to add and remove users dynamically from the Active Directory group controlling access to their application.

### 6.5.2 HTTP Headers

ClearTrust SSO works by storing the ClearTrust session token in an HTTP cookie. The browser returns this token as a message header in every request to a server in the same DNS domain as the server that originally issued the cookie. When a SecureControl Web Server Plug-in receives the cookie, it will decrypt it using the key it received from the Dispatcher/Key Server. The Web Server Plug-In then extracts from the token the identity of the user, how the user was authenticated, etc. If the user is allowed to see the content that was requested, then a new token, with an updated last touch time, is returned as a header in the HTTP response. For security reasons, the HTTP cookie specification allows a web server only to manipulate cookies that will be sent back to itself or to other servers in the same domain. This is why ClearTrust's traditional SSO does not work between domains. While a cookie set by one server will also be sent to other servers in the same domain, it will not be sent to servers in any other domain.

Below is an example header file extracted after a test user (tfwuser1) authenticated to an application web page.

```
Request Header: HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
HTTP_ACCEPT_LANGUAGE:en-us HTTP_CONNECTION:Keep-Alive
HTTP_HOST:TFWeb.spawar.navy.mil HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE
5.01; Windows NT 5.0) HTTP_COOKIE:ctrust-session-
v002b=k8oEpeyxlKDoGHAuN8a6V0sbqJPX1vwzXThKbNPC7v2JvkKuOBn2uQO3woy0BXW
vdAwP+cw4oz8lt3zp/ZrbhXYK/XkeyYMi3Ay6nyv1GQ/JE0nvS3NwUMHDbHG4Ck9HgOyQS
pIVwamYwJGg1xe5ftmgXYT6cy2dzEyVCHrQ1ZXdHQDFHm9jGUT0X9HVwKjo+o9Lgw/aNg
= HTTP_AUTHORIZATION:Basic dGZ3dXNlcjE6 HTTP_ACCEPT_ENCODING:gzip, deflate
HTTP_CTID:21 HTTP_CT_REMOTE_USER:tfwuser1 HTTP_CT_AUTH_TYPE:16
HTTP_CT_WEB_SVR_ID:WebServer
```

- The HTTP\_COOKIE field contains the ctrust-sessionID followed by the encrypted session cookie. This would normally be decrypted and read by the ClearTrust Web Plug-In to determine the user's userID and information about the current session.
- The HTTP\_CTID field contains a unique ID assigned to this particular HTTP header request.
- The HTTP\_CT\_REMOTE\_USER field contains the requesting user's userID. Application developers should key off of this field to compare it to their internal list of authorized users.
- The HTTP\_CT\_AUTH\_TYPE field contains a code representing the type of authentication used to identify the user.
- The HTTP\_CT\_WEB\_SVR\_ID field provides the ID name assigned to the web server upon which the Application is being accessed. This name comes from the ClearTrust Entitlements Database. The additional fields that ClearTrust adds to the HTTP header, such as HTTP\_CT\_REMOTE\_USER and HTTP\_CT\_AUTH\_TYPE are intended for use to extend the reach of the SSO solution. However, TFWeb recommends that these fields are not trusted for use as a means to authenticate users to an Application. They may be used to provision other non-sensitive services such as personalization of a page that does not require authentication. If an Application chooses to re-authenticate users, these fields may be ignored.

### 6.5.3 ClearTrust Web Server Plug-Ins

As described above in section 7.2.1, the ClearTrust Web Server Plug-In automates the process of identifying a user who has already authenticated to the ClearTrust SSO. The Plug-In could decrypt the session cookie to determine the userID of the individual accessing the Application's web server, but would require secure communications with the centralized Dispatcher/Key Servers to do so. Navy firewalls policies prevent these communications from occurring through firewalls. Therefore, in order for an Application to utilize the ClearTrust Web Server Plug-In to provide SSO within the Portal architecture, the Application would have to be hosted:

- Within the NMCI network,
- Within the Fleet NOC network,
- On the ship where the Portal resides.

The Application developer would still have the responsibility of programmatically assigning the correct rights to that user based on his or her new flat name space userID, as previously described. ClearTrust SecureControl provides APIs that give developers programmatic access to ClearTrust cookies and the Entitlements Database. Integration of this type gives greater control over the interface with the ClearTrust SSO system, and would expose the fields passed by the encrypted session cookies. If an Application developer chooses to pursue use of the ClearTrust Web Server Plug-In for SSO integration, the AMTS team will provide details on the ClearTrust APIs.

## 6.6 Limits to TFWeb Single Sign-On

The basic intention of ClearTrust's SSO solution is to eliminate the need to re-authenticate to web Applications any time a user accesses more than one portal Application in the same session. The hope is that the user only provides their credentials (userID and password) once to access the Portal, and will not be required to re-authenticate when using any of the portal Applications.

However, because ClearTrust, and most other web SSO products of its kind, use cookie technology to pass the user's authenticated identity from the user's browser to web servers, there is a known limitation to the SSO. The limit is the Internet domain boundary. That is, cookies originated in one domain (e.g., navy.mil) will not be passed to web servers in other domains (e.g., usmc.mil or any .com domain).

Therefore, if a user authenticates to the Portal in the navy.mil domain, and accesses an Application in the usmc.mil domain, they will be required to re-authenticate to that domain. However, once the user has authenticated in the second domain, he or she will not be required to authenticate in that domain a second time.

## 7 Application/Service Integration

The application integration process will differ depending upon the type of application to be integrated, and the level of integration the application will be achieving. In all cases the goal is to provide the developer a process and supporting infrastructure by which they can develop, test, and certify their application(s) for use in the TFWeb portal environment with a minimum involvement by a core TFWeb team or other external agencies.

This section describes the specific, technical integration issues involved with integrating an application into the Enterprise portal.

### 7.1 Assumptions and Limitations

The Enterprise portal will provide TFWeb information access for all DoN personnel. The TFWeb Portal will be a standards-based implementation where the method of support will differ based on bandwidth, security, and client posture used to access the portal environment.

#### 7.1.1 Standards Based Approach

The TFWeb Portal is to be based on standards wherever possible to minimize the impact of commercial product changes in the architecture and to simplify integration of existing capabilities.

Figure 7-1 highlights the important standards to be used by the TFWeb Portal at the appropriate level in the architecture.

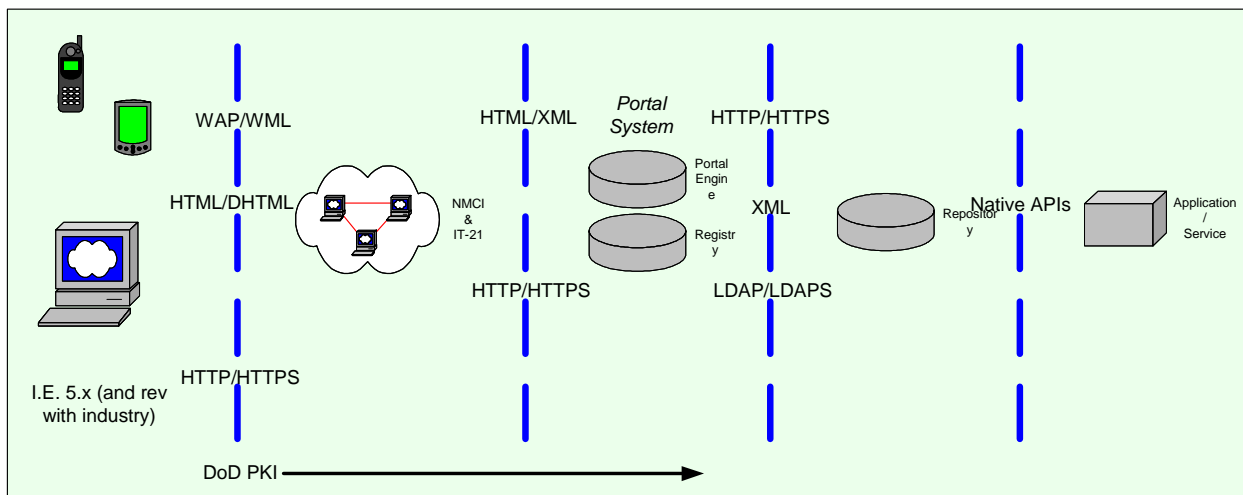


Figure 7-1 Standards Based Architecture

#### 7.1.2 Bandwidth Limitations

Replication of databases across the enterprise must be provided in an efficient manner that optimizes the use of limited bandwidth. This is required to prevent network saturation during periods of extended outages (e.g., EMCON, network outages) and allow prioritization of the flow of updated information. Deployers of TFWeb applications must provide a data flow model that will be evaluated by the WEN IT Governance Board-designated technical support team. The evaluation process will be for determination of bandwidth integration only. The process will be streamlined to accommodate simplified integration of applications into the WEN environment.



### 7.1.3 Security

Security within the TFWeb architecture is implemented using an “Access then Deny” approach. Each portal connector corresponds to a service within the service repository. All portal connectors are visible to all users within the portal. The URI associated with each service is protected by the TFWeb security infrastructure. Access is either allowed or denied when the portal connector attempts to access the URI associated with that service. It is the service owner's responsibility to provide the necessary authorization information to the Directory Services Administrator to set the access control list.

Please refer to Section 7 for additional details regarding security and authentication.

### 7.1.4 Service Module Definition

A service module provides connection between an application and the portal. The service module ensures modularity of the architecture by insulating applications from the portal.

The service module may be written in ASP, JSP and CGI (wintel C/C++ or Perl). The service module will no business logic for the application it is simply a connection to the application. There are 3 types of service module:

- ÷! Hyperlink Integration – Provides a hyperlink that opens a new browser window
- ÷! Presentation Integration – Application content is displayed through the portal pane.
- ÷! Application/Data Integration – Application communicates through TFWeb repository via Web Services like SOAP interface utilizing XML with is rendered in the portal using standard style sheets.

### 7.1.5 Local/Claimant vs. Enterprise Service Repository

A Local/Claimant Service Repository may be implemented for any of the following reasons

1. Firewall, bandwidth restrictions, or COI issues do not support connections from the Enterprise Service Repository to the application.
2. Specialized products need to be used that are not supported by the Enterprise Service Repository.
3. Connections to backend applications will take place through unsupported protocols. The Enterprise Service Repository will only support connection with HTTPS/SOAP/XML.

If a Local Service Repository is being used the service developer must:

1. Register the service with TFWeb service registry
2. Implement the standard PRI (Portal Repository Interface)
3. Support connectivity from the enterprise
4. Must handle errors in the same manner as the enterprise repository (e.g. 401.3, access request form)

## 7.2 Application Packaging Requirements

### 7.2.1 Application/Data Adaptor (Component Software)

The application/data adaptor is the software component that is created and maintained by the service provider and provides the connectivity to the legacy application(s). At a minimum, the TFWeb Portal will provide interoperability with the following native APIs:

- ÷! XML (desired objective)
- ÷! Java Server Pages (JSP)
- ÷! Active Server Pages (ASP)



÷! Common Gateway Interface (CGI) using the following languages:

- Perl
- C++

The application/data adaptor must output its result in XML (desired objective) or HTML format.

### 7.2.2 Service Registry Metadata Information

This section will describe an XML formatted data file that provides basic service information to the enterprise portal's service registry for identifying and accessing the specified service.

### 7.2.3 XML Schema

Each service provider is free to define and provide an XML Schema file to describe its application data structures. These XML Schemas shall follow the procedures defined in the DON Interim Policy On The Use Of Extensible Markup Language (XML) For Data Exchange, of 6 September 2001.

This policy directs that developers shall, to the maximum extent practical, make use of existing DoD and Industry XML components. In addition, if new XML components are developed they must be registered in the DoD COE XML Registry. Information about submitting your XML components to the COE Registry is available at <http://diides.ncr.disa.mil/smlreg/index.cfm>. The DON CIO has stood-up an XML Working Group, that is currently in the process of defining the policies, procedures, guidance, organizations, and infrastructure, which will be required for successful implementation of XML throughout the Department of the Navy (DON). Further direction on the use of XML throughout the DON will be provided, as the working group completes it's tasking.

## 7.3 Integration Examples

Before describing the APIs and details for integrating with the TFWeb Portal, the following subsections will describe some simple examples of each level of integration. These examples along with the repository of service adaptor components are intended to serve as templates that can be re-used as guides for your application integration to simplify the portal integration effort.

### 7.3.1 Hyperlink Integration

Hyperlink integration is the availability of a Hyperlink to the service or application from a page in the TFWeb Portal that transfers the presentation view from the TFWeb Portal to the selected application. The supplied Hyperlink must render a browser-based presentation in pure HTML. The application-supplied HTML will be rendered in a browser window with an appended portal "border" as described in Section 7.4.3. It is highly desirable that the application presentations conform as much as possible to the "look and feel" requirements as described in Section 7.4.3.

As an objective requirement for the service or application that is available via Hyperlink to the TFWeb Portal, the service or application needs to be able to accept individual user login information from the TFWeb Portal to automatically authenticate the portal user to the service or application.

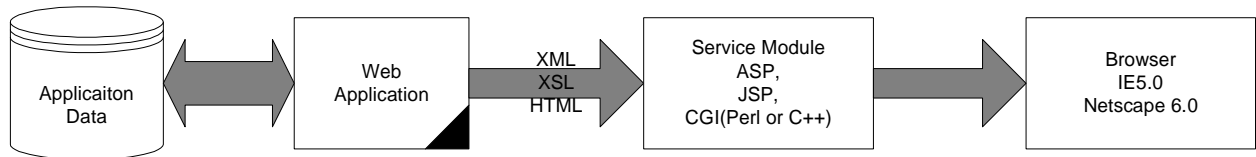
An example of a service module that implements hyperlink integration is contained in Appendix B.

### 7.3.2 Presentation Integration

Presentation integration makes data from a service or application visible through a user interface pane of the TFWeb Portal. For the TFWeb Portal, a pure HTML output from the application adaptor component will be acceptable in the near-term, but the desire is an XML output with its associated schema and XSL (style sheet for HTML translation).

In the examples below, the desired XML output is demonstrated. An application's data is stored in a database and made available through a, Java Server Page (JSP), Active Server Page (ASP), or CGI (Perl or C++) service module. The application data, once extracted, from the database is

formatted into an XML data file consistent with the application-defined XML Schema. The XSL style sheet is processed to render the XML data file into HTML for display in a pane of the TFWeb Portal. Figure 7-2 demonstrates the flow of data through from the legacy database to the browser



**Figure 7-2 Legacy Data Presentation Examples**

An example of a service module that implements presentation integration is contained in Appendix B.

### 7.3.3 Application/Data Integration

This section describes the basic implementation requirements for an Application/Data Integration service module. For a source code example of this please see appendix B.

#### 7.3.3.1 Parse PRI Request

The Portal to Repository Interface (PRI) will place the XML PRI request in the HTTP header. The service module must determine if the PRI request is present and if it is valid. Standard Java class are available as part of the repository to allow for this validation, see appendix B for a description of this class, PRIRequest. If the PRI request is not present in the HTTP header or the PRI request is invalid, the service module must exit with a 403 error.

#### 7.3.3.2 Make Application Request

Two types of requests can be made to the application for data. The preferred method is to make a SOAP request. By the nature of the request SOAP an XML document will be return in the response. Another method for the request is an HTTP request. An HTTP request can return both HTML and XML as a reply, the preferred method is to return XML whenever possible.

#### 7.3.3.3 Application Accepts Request

This is not part of the service modules process. The application that a request is made of, as described in Section 7.3.3.2, will have to accept and process the request. The response with either be a SOAP or HTML.

#### 7.3.3.4 PRI Response

The service module will create a PRIResponse object (see appendix B for description). The PRI Response will be in the HTTP header for processing by the Portal Repository Interface (PRI).

#### 7.3.3.5 Application Results Return

Lastly, the service module will return the results from the application request and the PRI Response to the portal. The portal will format this data for rendering within the browser.

## 7.4 Messaging Protocols

A messaging transport is the technology that facilitates peer-to-peer application communication using open standards. The repository will communicate with applications via the following protocols:

÷! HTTP

- ÷! HTTPS

- ÷! SOAP

## 7.5 Management

This section will provide implementation details pertaining to the Enterprise portal components that will be of interest to service providers in integrating their service components.

### 7.5.1 Service Registry

The service registry is where the metadata for the component modules is held--for each module that's been created and published for end-user access within the Enterprise Portal. It includes such information as:

- ÷! Source applications

- ÷! Database connections

- ÷! General Access Rules

- ÷! Delivery and Presentation Requirements (e.g., XML/XSL)

In general, it is the developer's responsibility to provide metadata content to the TFWeb portal administrator for inclusion in the registry. Most portal engines are capable of automated metadata extraction to populate the service registry via the application or data object publish process. It is a goal of TFWeb to provide some mechanism to enable the developer to integrate into the portal (and therefore, populate the registry) without direct intervention of the Portal Administrator. Service registry population processes (to include metadata data/packaging formats) will be delineated once portal engine selection is accomplished (e.g., vendor-specific instructions).

### 7.5.2 Component Repository

Architecturally, a distributed set of application servers will provide the support infrastructure for Enterprise Portal components. While a wide variety of products advertise themselves as being "application servers," the two dominant "standards" are Sun Microsystems' Java 2 Enterprise Edition (J2EE) and Microsoft's "DNA" (Distributed Internet Applications) framework. Microsoft's ".NET" architecture is due to be released sometime in 2001, and will add to DNA a set of language-neutral libraries and development tools. Both standards offer commercially acceptable solutions to implement web applications and services. Both J2EE and DNA/.NET provide an "object model" that defines how components should be constructed and a set of roughly equivalent "services" that can be accessed by the components.

The repository will provide the following services to the Enterprise Portal:

- ÷! Component storage.

- ÷! Component and content connection management/pooling.

- ÷! Component naming.

- ÷! Content validation and parsing.

- ÷! Component deployment.

In general, it is the developer's responsibility to provide the components and content to the TFWeb portal administrator for inclusion in the repository. Most portal engines are capable of automated component/content extraction to populate the service registry via the application or data object publish process. It is a goal of TFWeb to provide some mechanism to enable the developer to integrate into the portal (and therefore, populate the repository) without direct intervention of the Portal Administrator. However, the TFWeb technical team is responsible for mitigating inter-component conflicts. Service repository population processes (to include

data/packaging formats) will be delineated once portal engine selection is accomplished (e.g., vendor-specific instructions).

### 7.5.3 Session Management

Session management must be accomplished via the portal vendor's session management modules/APIs, which will include:

- ÷! Maintaining session state by enforcing access control rules and timeout as needed.
- ÷! Provides active session state visibility/currency by maintaining peer sessions between portal web server applications and end-users.
- ÷! Performs dynamic load balancing by either having a front-end engine that balances the components within the portal engine, or by internal parallel processing capabilities (dependent upon vendor).

While the developer does not have any specific requirements to provide data or metrics for this environment, information about session state may be obtained from the portal engine. The developer, via their application component, can provide the session management engine with specific session data (e.g., session idle timeout, destroy/reactivation time limits). Session state data processes (e.g., APIs, wizards) will be delineated once portal engine selection is accomplished (e.g., vendor-specific instructions).

### 7.5.4 Replication

The data repository environment will provide scalable replication services. These services will enable the content provider to persist the data throughout the enterprise as determined by the agency/application manager's business rules.

The program, application, or content manager will determine the replication requirements, XML schemas, and periodicity to the developer and the Enterprise Portal team (via checklist). The TFWeb will act as arbitrator/negotiator between overlapping data distribution requirements. The developer will use the Enterprise Portal replication services for data distribution.

Data replication processes/interfaces (e.g., APIs, protocols) will be delineated once data replication engine selection is accomplished (e.g., vendor-specific instructions).

## 8 Service Certification Process

### 8.1 Process Overview

The TFWeb Service Certification Process is designed to ensure application services meet the security and functional standards of TFWeb and the Government prior to implementation within the production TFWeb Portal. Figure 8-1 Service Certification Process.

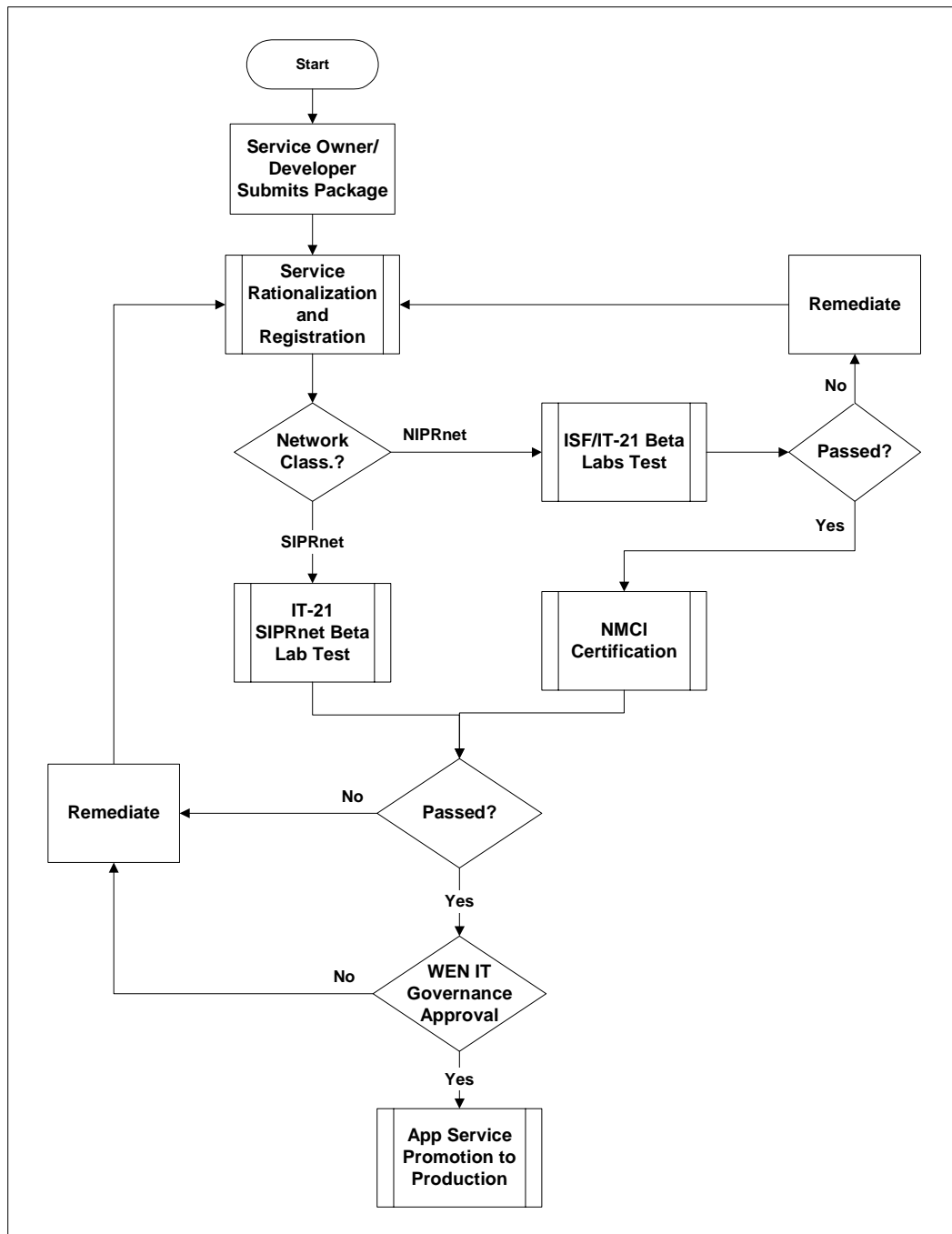


Figure 8-1 Service Certification Process.

The certification process commences when the Service Owner delivers the Intent to Migrate Package to its Application Migration Customer Support (AMCS) point of contact (POC). The AMCS Team will review the Intent to Migrate Package for completeness and assists in assembling the required information for approval of the final Request to Migrate by the AMCS.

The TFWeb Beta Test Team will then perform Beta Testing within both the IT-21 and NMCI Beta Lab environments. The Engineering Team will work with the Service Owner to address any issues discovered during the portal integration testing. Any service that will operate across the NIPRNET will need to pass through the NMCI Application Certification process to gain approval to function across the NMCI network firewall boundary.

## 8.2 Information Assurance Certification and Accreditation

It is the Service Owner responsibility to obtain an ATO or IATO for an application prior to registering it for migration to the TFWeb Portal. Please see the ISSM representative for your command for more information.

## 8.3 Service Intent to Migrate, Rationalization, and Registration

The Intent to Migrate package will consist of a submission to the application information database managed by AMCS and an IATO/ATO if one currently exists. The package will be reviewed by a member of AMCS using the guidelines in section 12.2 to determine the requirements for the final Service Registration package.

The Service Registration package should be submitted by an authorized representative of the Echelon II command to the appropriate AMCS POC via email and should address the issues listed below. Portions of existing documents may be submitted as part of the memo to prevent unnecessary duplication of effort, however the submission should be organized to provide the following information. Guidance is provided in section 11.3 on the Service Registration package review process

- ÷! Interim Authority to Operate (IATO) or Authority to Operate (ATO) from the appropriate Designated Approval Authority (DAA) for the software developer.
- ÷! NMCI Request for Service (RFS) form for NIPRNET applications
- ÷! Migration plan to level 3 integration with appropriate milestones (separate document)
- ÷! Waiver for level 1 integration included (if applicable)
- ÷! Registry Meta-data (section 8.2.1)
- ÷! Service Repository package (section 8.2.2)
- ÷! Access control list (section 6.5.1)
- ÷! Test plan and cases
- ÷! Temporary login with access to non-administrator portions of the application - if a level 1 or 2 application
- ÷! Summary of previous testing accomplished
- ÷! Configuration of local application servers or remote module servers and estimated concurrent users of service
- ÷! Documentation of application data structures and data interfaces
- ÷! Migration plan - if application/data overlap has been identified
- ÷! Migration plan - if XML not in compliance with Navy standards is in use

### 8.3.1 Registry Metadata

The Service Owner should include the following information to be integrated in the portal registry.

- ÷! Description of portal service to be integrated
- ÷! The URL of the service to be integrated
- ÷! The Owner of the service to be integrated (lead development organization)
- ÷! The taxonomy category under which the service will be listed.
- ÷! Point(s) of contact information for user access. This information should include names, phone numbers, email addresses
- ÷! Any parameter information required by the service
- ÷! Target User Community (role/platform/location)
- ÷! Service versioning information

### 8.3.2 Service Repository Package

The Service Owner will need to put together a service repository package that includes any resources the service requires being included in the service repository. The service repository package contents will vary depending on the level of integration required by the service.

These service repository resources can include any of the following types of items: HTML pages, icons and images, XML files, XSL templates, JSP pages, Java Servlets, EJBs, ASP pages, COM/COM+ components.

The following file formats are acceptable for the service repository packages:

- ÷! Java developers should deliver their service repository package in WAR/EAR format. WAR/EAR files are an executable file archive format used to package deployment files for a Java enabled application server.
- ÷! ASP developers should deliver their repository package in CAB format. The CAB format is a file archive pattern used to package deployment files in the Microsoft Information Server environment.

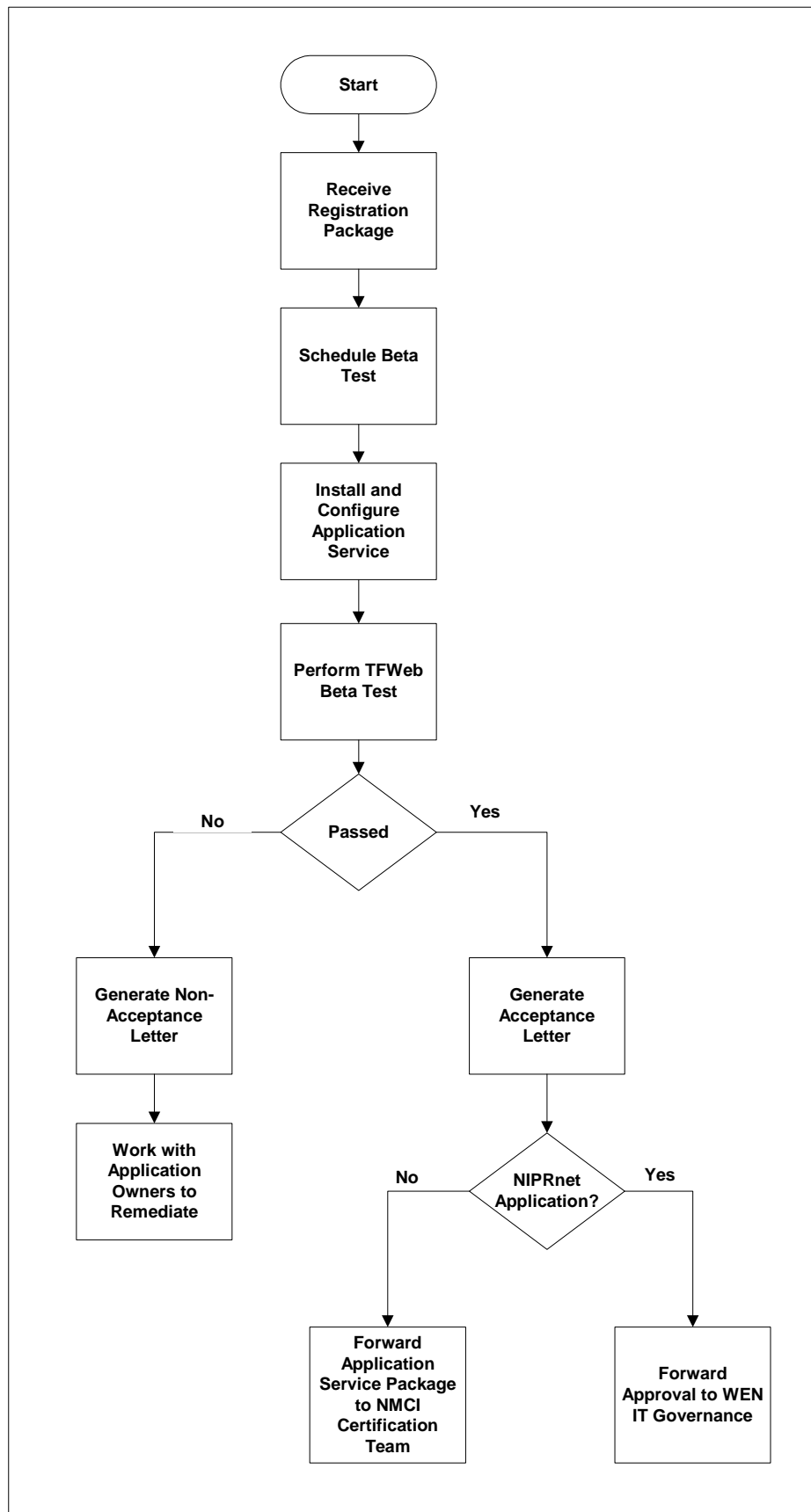
Other developers should deliver their service repository package in ZIP format. ZIP is a multipurpose file archive format

## 8.4 TFWeb Beta Lab Test

The TFWeb Beta Lab ensures that application services function appropriately within the portal environment and that they adhere to the TFWeb standards outlined in the Integration Developers Guide. There are two TFWeb Beta Test Labs, one for IT-21 and one for ISF. All application services will be tested in both the IT-21 and the ISF Beta Labs.

**Figure 8-2 TFWeb Beta Test Process**





### 8.4.1 Process

1. The AMCS POC submits service package to Beta Lab POC via email. The package will include the following items:
  - ÷! Certification of Ready to Migrate
  - ÷! Registry Meta-data
  - ÷! Service module package
  - ÷! Access control list
  - ÷! Test plan and test cases
  - ÷! List of flat name space id of users to whom access should be granted through the service module
  - ÷! Description of any special application functionality that will be required and/or tested
2. The Beta Test Team schedules a time for testing of the application service module.
3. The Beta Test Team creates the test plan, incorporating the test cases submitted with the package.
4. The Beta Test Team installs and configures the service module and portal connector.
5. The Beta Test Team performs the tests.

Below are some of the criteria against which applications will be tested in the Beta Lab.

- ÷! Use of standard protocols between the service module and the service.
- ÷! Conformance to the TFWeb Look and Feel policies and guidelines.
  - ÷! Presentation should support the appropriate media formats (e.g., graphics, audio, animation, compression, etc.).
- ÷! Functionality
  - ÷! Standard HTTP error return codes are returned in case of an exceptional or abnormal condition.
  - ÷! Handling of pop-up windows
- ÷! Conformance to TFWeb Security guidelines

To participate in the overall Single Sign-on solution, application developer's must:

- ÷! Read and trust the authenticated user ID from the TFWeb HTTP header.
  - ÷! Map TFWeb user IDs to the application's Internet user rights database or develop a user rights database using the TFWeb user ID.
  - ÷! Authorize user's rights to internal application processes or databases based on the authenticated user ID provided by ClearTrust.
6. If the application fails any test cases or if its performance impacts that of the Portal environment the application will not pass the Beta Test. In this case the AMCS POC and the Service Owner are notified with specific reasons for failure. AMCS may request support from AMTS to remediate any technical issues preventing approval.
  7. If the application passes the Beta Test an approval letter will be sent to the AMCS POC and Service Owner.
  8. Once approved, the service will progress to the WEN IT Governance if it is a SIPRnet application. If the service will function across NIPRnet it must first go through the NMCI

Certification process before going to WEN IT Governance. After WEN IT Governance the service is promoted to production.

## 8.5 NMCI Application Certification Process

The NMCI Certification Team must certify services that will function across NIPRnet. The objective of this certification is twofold. First it ensures that the application service will function correctly in the NMCI environment. Secondly it ensures that the communication across the NMCI boundary firewall is secure and any inherent risk with the communication is understood and accepted by the Government.

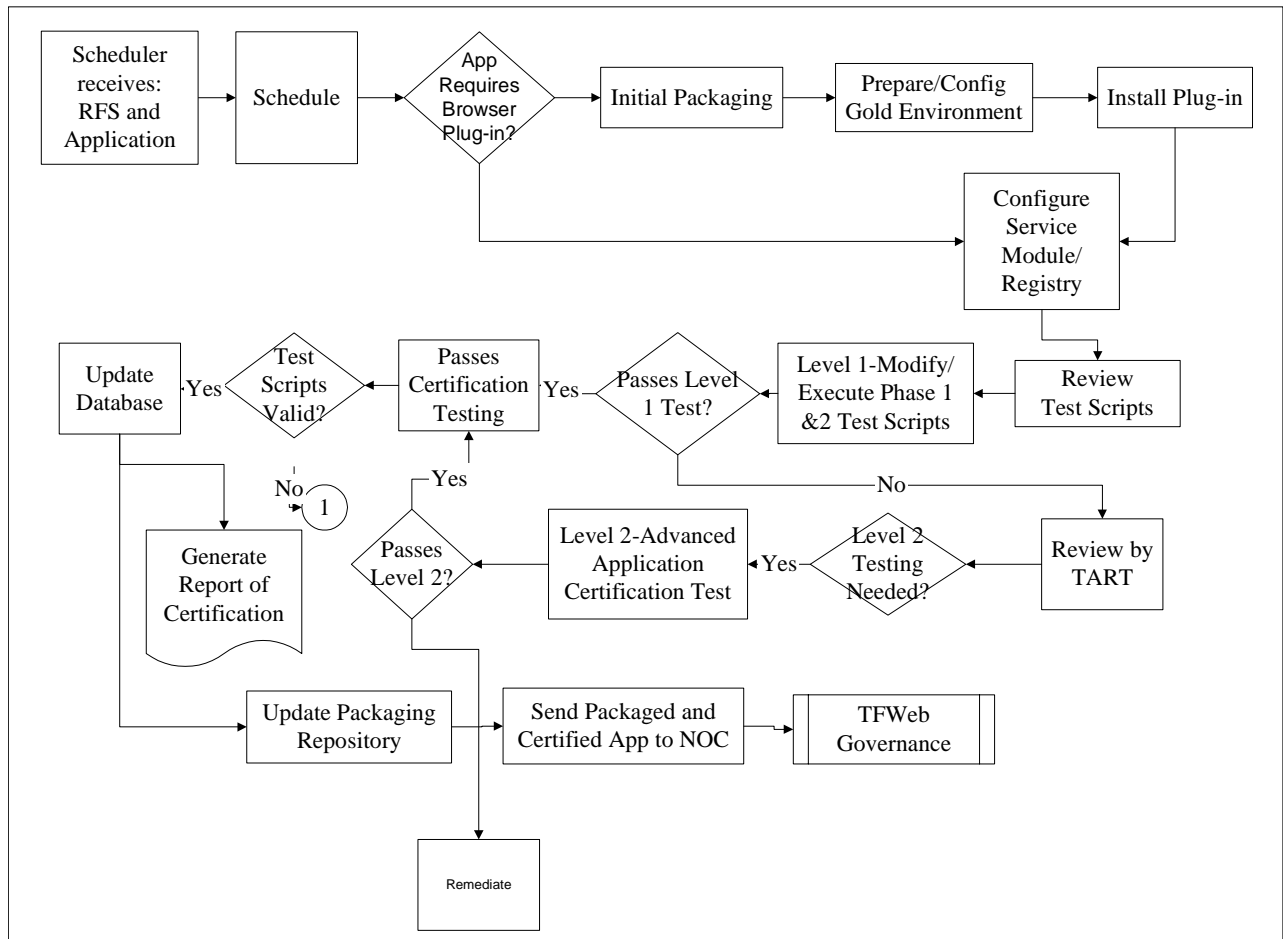


Figure 8-3 NMCI Certification Lab Process

### 8.5.1 Process

1. The AMCS POC submits certification package to Certification Lab Scheduler. The package will include the following items:
  - ÷! NMCI ISF Request for Service (RFS) form
  - ÷! Package submitted to TFWeb Beta Lab (see Section 8.4.1)
2. Certification Lab Scheduler reviews package for completeness
3. If Package is incomplete it is returned to the AMCS POC for completion. The AMCS POC may be required to contact the Application Owner for additional input or revision.

4. If Package is complete the Certification Lab Scheduler reserves a time slot for testing of the application service module.
5. The Certification Lab Scheduler forwards the certification package to the Certification Team.
6. The Certification Team creates the certification test plan, incorporating the test cases submitted with the package.
7. The Certification Team installs the web service module from the media and configures the Portal and Repository module according to the information from the Service Registration Form and the TFWeb Developer Guide (contains standard installation instructions).
8. The Certification Team executes the tests. The Certification Team will be responsible for processing all applications through a two-phase testing cycle. Phase 1 consists of basic application functionality testing, and Phase 2 consists of standard seat service integrity testing. Phase II is only required if a service requires browser plug-ins that have not already by certified for deployment into the NMCI environment. Any special or additional test requirements must be identified by the Navy claimants prior to the beginning of testing, and preferably in the NMCI RFS submitted to the Beta Lab to initiate testing.

The certification criteria currently required for Phase 1 testing include:

- ÷! Basic functionality:
- ÷! Launch the application
- ÷! Create or Open a new document or file
- ÷! Save a new document or file
- ÷! Print a new document or file
- ÷! Close the application

Any additional functionality testing needs to be defined and scripted by the customer to ensure proper execution

Once the NMCI Beta Lab receives an application, the site or customer can track the status by viewing the report posted online at <http://www.eds.com/nmci/transition.htm>.

9. If the application fails any test cases or if its performance impacts that of the Portal environment the application will not pass certification. In this case the AMCS POC and Service Owner are notified with specific reasons for failure.
10. If the application passes a certification approval letter will be sent to the AMCS POC and the Service Owner.
11. Once certified the application is then reviewed by the WEN IT Governance for approval for promotion to the production portal.

## 8.6 WEN IT Governance

Programs that do not meet all requirements for migration may rarely be allowed to proceed through the testing process while simultaneously completing these requirements. In addition, applications that fail portions of the testing may be functionally displaced by another application by the time they are ready for migration to the production portal. Testing may also demonstrate substantial overlap with another application or organizational issues that prevent immediate migration of the application. Final approval of migration is currently a function of the Task Force Web Executive Steering Group. This approval may be delegated to a lower level based on application compliance with TFW standards.

## 9 Coding Standards, Policies and Guidelines

In order to maintain a large organization of applications and services, certain code based naming conventions need to be applied. This guide is presented as a straightforward suggestion that will streamline potential conflicts within each service and application. This is only meant to be a guideline where there are no guidelines present. Where there are current guidelines, those should take precedence over any procedures in this guide. Many of these suggestions have been adapted as the “industry standard” or “best of breed” and are well known within the IT industry.

Questions and suggestions should be referred to the NMCI System Administrator for clarification or for reference material.

### 9.1 Directory Structure and Variable Naming Conventions

While the directory structure for a service module is highly subjective to the internal plans of the particular development shop, it was felt necessary to provide some guidelines on directory structure and variable management. Below is a suggested outline for application directories. In the next section variable management will be discussed. This should not supersede any internal mandate already in place.

The use of this outline is suggested and voluntary. This section was written to aid in the development process for the service application provider. Any questions regarding the subject matter below should be directed to the NMCI system administrator.

#### 9.1.1 Assumptions

- ÷! The web administrator or web master has provided space for the web site on the web server
- ÷! All virtual directories have been created and configured by the web master or web administrator on the web server
- ÷! The user has sufficient rights to add, edit and delete files and directories in the target environment
- ÷! A saved backup incase a restore is required
- ÷! A clear idea of the functionality that is to be represented by the web site
- ÷! Source code control procedures or applications are being used

#### 9.1.2 Directory Structure

The directory structure listed below is a sample of a “best of breed” structure that is supported by many organizations throughout the Internet industry. One thing to be aware of is that there are no single files in the main directory. This helps modularize the development of code based on physical functionality. Consequently, this will aid in the maintenance of the web site as new ideas and functionality is implemented. There are other practices that allow for modularization based on functional requirements but code reuse is not at a premium.

Below is a suggested directory structure the actual structure will need to be determined by the development team based on the service module requirements.

Service Key (usually the name of the website)

- ÷! Cgi-bin – this directory is used for any scripts or executables that may be executed in the application
- ÷! Images – all graphics necessary in the view of the web site should be put in a single directory to aid with graphics management.

- ÷! Includes (optional)- any files that are reused in the web site
- ÷! Html – all html files
- ÷! Js (optional)- all javascript files that are reused
- ÷! Css – any style sheets used for the web site
- ÷! Etc (any specialized directory settings that aid in the development of the web site)

The Service Key, identified above, is a globally unique, 32-character hexadecimal value assigned to each service module as it is registered within the TFWeb Service Registry. The Service Registry administrator performs registration of service modules. Registration occurs following successful certification of the service module. During the development phase, it is recommended that the Service Key directory be named in relation to the functionality being provided by the service module.

### 9.1.3 Filename Standards

General filename standards should also be present when developing web sites. Each file should tell a little bit about what the file should do. This helps developers to organize code in a way that is logical and somewhat organized. As always the shop rules apply to filename naming conventions before applying any outside rules.

Each file should be saved in its appropriate directory, with the appropriate extension, in order to promote organization and reuse.

## 9.2 Variable Management

Variable management is also another aspect of web site planning that is highly subjective to each shop. If the current shop has already published guidelines for variable management, all parts of this discussion that comply with the policy should be followed, noncompliant directions should not be followed. This discussion is for service developer's information and should be taken into account when using client side variables.

By complying with the broad guidelines below, "variable collision" could be held to a minimum. "Variable collision" can be defined as two variables with the same name that have different functionality within applications. The collision occurs when the variable is called and the desired functionality does not occur, other functionality has rendered the variable inconsistent with the desired results. This behavior can occur when using cookies, javascript or other client side validation techniques.

### 9.2.1 General Naming Conventions for Variables

In general naming conventions should be meaningful to the web site developer and should describe the functionality of that specific variable. As with any programming language, any variables should be named to express function or purpose. Care should also be taken to not use reserve words as variables because there could be unexpected results. When appropriate the developer should comment the application to aid with maintenance issues.

Comments help to explain why and how this part of the code works. This allows for more detailed documentation right where the developer needs it, in the code. There are many different ways to comment code a standard should be defined and followed throughout the coding effort. Check with the particular programming language to detail how to comment functionality within the code.

### 9.2.2 Local Variables

Local variables are variables that reside inside a function or procedure. These variables should not have subsequent pages rely on the values, as they will disappear on any subsequent page. Nonetheless, local variable naming should also express function or purpose. When necessary, it

is always a good idea to type cast and declare variables (dim persarray(9) as array, declare persarray[] as array).

The following are examples of good variable names.

- ÷! personcount – counter to increment number of people logged in.
- ÷! lname - last name
- ÷! fname – first name
- ÷! rank – rank

Some examples of inefficient variable names:

- ÷! X
- ÷! Y
- ÷! Ddrfdse -unless it makes sense
- ÷! Yadayadayada – not descriptive enough

### 9.2.3 Global Variables

Global variables should be avoided if possible. If a global variable is used, make sure the variable is prefaced with some indicator that it is globally unique. A good naming standard is one that is planned in advance. This will also aid in the “non-collisionary” variable path that each web site seeks to encounter. Once a global variable is not used, destroy it so as not to cumber other application specific functions.

The following are some examples of global variables:

- ÷! gblUserID – the gbl designates that the UserID is global
- ÷! gv\_Role – gv\_ designates that the Role is a global variable
- ÷! globalRank – global designates the scope of the variable

Some poorly defined global variables:

- ÷! Out – could be confused with other functions like print.out
- ÷! In – could be confused with other functions like input()
- ÷! Count – reserved word
- ÷! Id- could be confused with any other id that may be used

### 9.2.4 Permanent Client Side Cookies

The use of cookies should be restricted as the cookie can always be tracked back to users or user computers. DoN CIO has found that cookies are in violation of a federal policy that prohibits the use of Internet technology that collects identifying information on individuals who access its web sites. That policy prohibits the use of web technology to collect identifying information to build profiles on individuals, and prohibits the use of persistent cookies unless certain conditions are met, including obtaining the personal approval of the head of the agency.

If a cookie must be used, use the GID number. This a unique number for each web site on the NMCI web portal. This unique naming convention will almost guarantee that cookies are not over written.



### 9.2.5 Server Side Session Variables

Server side session variables should use the same naming convention as all other variables. As always the shop conventions should be adhered to before changing any parts of the code. Session side variables should be used sparingly as it takes up memory on the server and could potentially cause lags in service.

In order to save some of the processing power, be sure to destroy all unused session variables at the time the session variable is no longer used.

## 9.3 Standard Error Trapping

Each component of the TFWeb infrastructure could generate an error during processing. Similar error could be generate by each component. For example, the Portal, the backend application and application connector could all generate 403 errors. If the error message that is returned to the user is a 403 error, it will be difficult for the help desk to determine the nature of the error and recommend solutions. Because of this situation a standard error trapping mechanism will be determined during beta testing.

## 9.4 JSP Standards

Java Server Pages (JSP) provides a great deal of flexibility and can become a challenge to keep package and naming standards consistent. In order to keep naming conflicts, memory leaks and database connections at a minimum, a standard approach to JSP must be taken. Many of these suggestions have been adapted as the "industry standard" or "best of breed" and are well known within the IT industry.

Questions and suggestions should be referred to the NMCI System Administrator for clarification or for reference material.

### 9.4.1 JSP Error/Exception Handling

For JSPs all errors should be directed through a common error handler, this insures that all errors will addressed in a similar manner with the user receiving a standard message. The following technique describes how this can be accomplished. Create the ExceptionHandler.jsp page. The following code should be used to create this page.

```
<%@ page isErrorPage="true" import="java.io.*" %>
<html>
<head>
    <title>Exception Occurred</title>
    <style>
        body, p { font-family:Tahoma; font-size:10pt; padding-left:30; }
        pre {font-size:8pt; }
    </style>
</head>
<body>

<%-- Exception Handler --%>
<font color="red">
<%= exception.toString() %><br>
```

```
</font>

<%
out.println("<--");
String Writer sw = new StringWriter();
PrintWriter pw = new PrintWriter(sw);
exception.printStackTrace(pw);
out.print(sw);
sw.close();
ps.close();
out.println("——>");
%>
</body>
</html>
```

The JSP should use the following construct to invoke the above error handler.

```
<% page errorPage="ExceptionHandler.jsp" %>
```

This would allow the exception handler to be invoked if any error occurs during the execution of the JSP.

### 9.4.2 Environment Cleanup

Environment cleanup refers to cleaning up variables, record sets, objects, connections, streams after you are done with them. As each object is no longer being used, it is a good idea to destroy these objects to save memory leaks and to have the application perform at an optimum level. Do not rely on the garbage collector to clean up the environment. It is up to each developer to make sure that his or her environment is optimal.

## 9.5 CGI Standards

Much like JSP, the CGI environment must be managed. Any questions regarding the CGI standards should be directed to the NMCI Support personnel assigned to your shop.

### 9.5.1 CGI Error/Exception Handling

When handling errors the first consideration should be that a user should never see a severe error. If a programming error occurs, the framework should trap it and send a detailed error page to the browser, not leave you staring at a "Server error" page in your browser.

#### 9.5.1.1 C

This area will be determined during the pilot.

#### 9.5.1.2 Perl

This area will be determined during the pilot.

### 9.5.2 Environment Cleanup

This area will be determined during the pilot.

## 9.6 ASP Standards

ASP coders should follow the Microsoft recommendations listed at the web site <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnasp/html/asptips.asp>. These recommendations follow a large number of topics that are too detailed to contain in this document.

### 9.6.1 Network Bandwidth Issues

1. In most cases network performance can ruin the ASP performance. Use 10/100 network cards for better performance.
2. If your Web server and the database server are running in the same server, then it is advisable to move them into different servers.
3. Even you can introduce a new middle-tier server to handle the COM components with MTS. This decision totally depends on the load on the Web and MTS servers.

### 9.6.2 Error/Exception Handling

Error and exception handling should be handled in a standard way. There are many competing schools of thought on exactly how to do this. One thing is for sure, is that there needs to be a standard way to process exceptions.

A streamlined error routine is to declare string constants with the error message and call those string constants when necessary. This will give you a single set of messages that you can use over and over again for different scenarios. Along with this approach, you can have a generic error page that displays the error text on it with varying degrees of functionality behind it. For instance, after an error the user should be able to go back to the previous page or be able to correct some of the submitted information. Having one standard error page that is called on every error will help you modularize functionality within the site.

To minimize client errors, it is usually a good idea to have some client side editing capability so that incorrect submittals are not submitted to the server for processing. Usually there are some client side validation techniques that will allow the user to correct the data before it is submitted to the web server. Sample techniques are to check the length of the submitted field to make sure there is data in it. A simple alert can warn the user that they will need to input some necessary data into a submitted field.

### 9.6.3 Environment Cleanup

In the case of HTML and ASP, one should always use environment cleanup. Don't assume all requests will be terminated and all database connections will be cut off. Do it your self. That is the most efficient way to keep the web server optimized and available for the next client to use the application.

## 10 Development Tools and Resources

### 10.1 Development Tools

For the purposes of Task Force Web (TFWeb), developer tools that directly correspond to TFWeb portal implementation are defined as follows:

- ÷! XML
  - Editors
  - Data Modeling
- ÷! HTML Editors
- ÷! Application development environments (e.g., MS Visual Studio, ANSI C)

As stated above, the purpose of this document is to provide guidance to service providers to enable easy integration of their applications into the Enterprise portal infrastructure. This guidance does not presume to specifically dictate a set of utilities that developers must use to develop applications.

However, there are a number of incompatibilities that many developers may encounter when using a certain toolset, which may preclude its use in the TFWeb portal environment. As such, the main Task Force Web portal site will provide a collaborative knowledge area devoted to the developer virtual interest group. This knowledge area will have the following sections:

- ÷! Preferred "Not to Use" Product List: This area will be devoted to identifying the products that have been found to be incompatible with the TFWeb portal infrastructure.
- ÷! Discussion areas, by product, in which various developers (and possibly their vendors) can discuss experiences with the toolsets. In this manner, developers and content providers can share their experiences/solutions gleaned from developing

Refer to appendix C for a complete list of the supported versions for each technology used by TFWeb.

### 10.2 Testing Tools

#### 10.2.1 Portal Connector Stub (PCS)

Portal Connector Stub (PCS) comprises of simple web pages that simulate the behavior of the Portal Connector Template. It allows the application developers to test out the service interfaces without needing to install the Portal Server at their development sites.

Developers will use a web server and the PCS at their site. The application owners develop and test their application interfaces with PCS following which they submit the service for integration testing and certification.

The PCS is developed using Hyper Text Markup Language (HTML), JavaScript and Active Server Pages (ASP). This tool allows the service developer to test the service by simply typing in the service URL in the provided text box and launching the service. PCS passes the necessary PRI (Portal to Repository Interface) data in the Request Header sent to the Service. The service will then extract the information from the Request Header variable PRIDataRequest and use it appropriately. The HTML or eXtensible Markup Language (XML)+ eXtensible Stylesheet Language (XSL) returned by the service will then be rendered in the frame inside the PCS tool.

PCS is a 'test' stub and not the real Portal Connector. PCS's purpose is to allow the Service Developer's to test the service without being required to install CA Jasmine Portal Server. Portal Connector Template is used at to create the actual 'Portal Connectors' for the portal to make the service available in the portal. In Portal Connector Template we will be certainly converting the

XML+XSL to HTML on the 'server side'. To keep the PCS simple, it will rely on the browser's capability for the XML to HTML conversion. Thus the browser used for testing the service MUST be able to handle XML.

To ensure proper testing, the future release of the Portal Connector Stub may include a XSL parser that will be used in the Portal by the Portal Connector Template.

### 10.2.1.1 PCS Layout

The look and feel of PCS mimics the portal to a certain extent. Main elements in the layout are as:

- **Header** – Consists of a header image from the Portal.
- **Control** – Consists of various control elements in the PCS outlined below:
  - **Service URL Box** – URL of the Service being tested is entered here.
  - **Go Button** – When Clicked sends a request to the URL of the service.
  - **Submit Option** – Get or Post Option allows the developer to choose how the form is submitted to call the service being tested.
  - **Clear Button** – When Clicked clears the Contents in the Content Frame.
  - **Content Layout Pull Down Menu** – To Select the layout of the Content Frame.
  - **Example Button** – When Clicked loads the content frame with an example image to give an idea to the service developer how different layouts look.
  - **Help Button** – When Clicked loads the help file in a new browser window.
- **Content** – Displays the response from the service.
- **Footer** – Consists of a footer image from the Portal.

Screen shots of the PCS are included in the following pages with different layouts for reference:

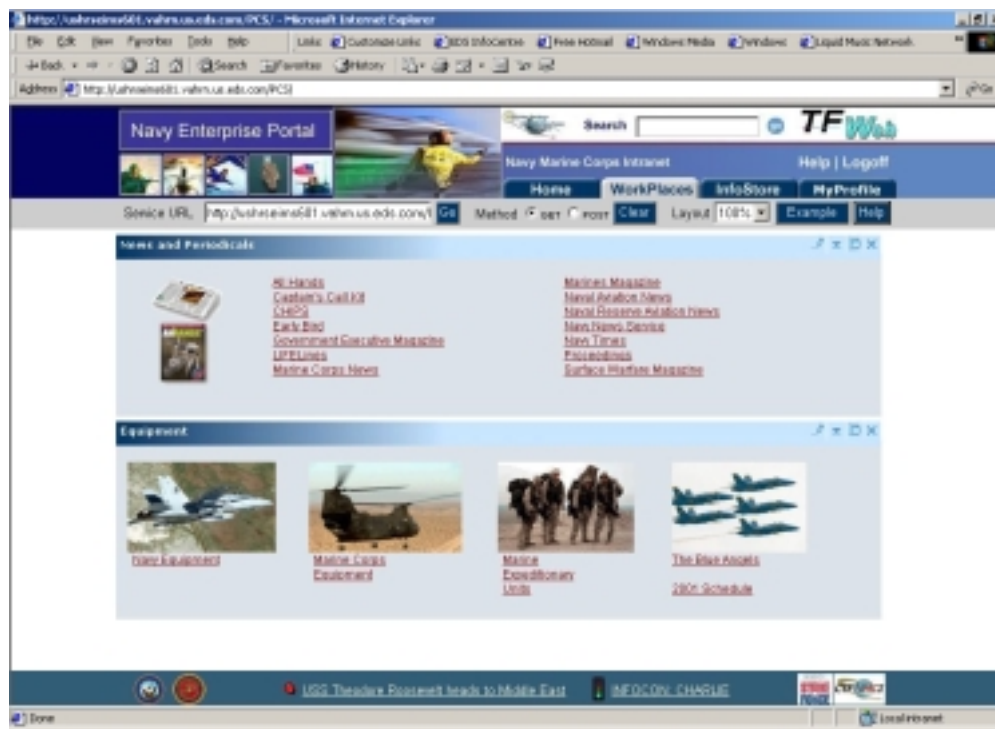


Figure 10-1: Content Frame Layout (100%)



Figure 10-2: Content Frame Layout (50:50)



Figure 10-3: Content Frame Layout (30:70)

The portal connector stub is designed for 800 X 600 and 1024 X 768 pixel resolution. For best presentation of the PCS screen resolution of 800 X 600 is recommended.

### 10.2.1.2 PCS Components

Folder Structure for the PCS Tool:

Portal Connector Stub

```
|
|---- Images
|
|---- PCSDData
|
|---- Test Service
```

- Portal Connector Stub Folder: Contains HTML files, and ASP file(s).
- Test Service: Contains test services
- Images folder: Contains all the image files used by PCS.
- PCSDData: Contains the XML Data Files (Input and Output)

Files that make up for the PCS Tool are illustrated below:

#### 10.2.1.2.1 System Files (Non-Modifiable)

##### ÷! HTML Files

- **default.html** – Entry Point of the PCS
- **header.html** – Creates the Header using an image.
- **controls.html** – Creates the Control area of the PCS
- **main.html** – Target for Response from Service when the selected layout is divided into one frame in the Content Area.
- **50\_50.html** – Creates 50:50 frames in the Content Area.
- **30\_70.html** – Creates 30:70 frames in the Content Area.
- **left.html** – Target for Response from Service when the selected layout is divided into two or more frames in the Content Area.
- **right.html** – Place holder in the Content Area when layout of three frames is chosen.
- **footer.html** – Creates the Footer using an image.
- **wholepage\_example.html** – Example Image for whole page layout
- **50\_50\_left\_example.html** – Left Example Image for 50:50 page layout
- **50\_50\_right\_example.html** – Right Example Image for 50:50 page layout
- **30\_70\_left\_example.html** – Left Example Image for 30:70 page layout
- **30\_70\_right\_example.html** – Right Example Image for 30:70 page layout
- **PCSHelp.html** – Help File for the PCS Tool

##### ÷! ASP File

- **Submit.asp** – Used to populate the PRIDData in the Request Header's PRIDData field.

##### ÷! Image Files

All Image files used for the PCS are located in the images folder.

##### ÷! Test Service

- **Test.asp** – This is a demo service that extracts data sent in the Request Header and renders it in the browser. Developers can use this to check the correctness of their installation of the PCS tool.



### 10.2.1.3 Data Files

The Data files exist in the 'PCSDData' folder under the 'PortalConnectorStub' folder. This folder MUST be shared with proper permission as 'PCSDData' in order to allow the PCS tool to read and write the XML data files.

#### 10.2.1.3.1 Input File(Modifiable)

- ÷! **PRIDataRequest.xml** This is an XML file, which consists of variables and their values as specified in the PRI. The variable information can be manipulated by the Developer to test services.

#### 10.2.1.3.2 Output File

- ÷! **PRIDataResponse.xml** This is an XML file, which consists of the XML data as per PRI interface, sent in the Response Header by the Service.

### 10.2.1.4 How To Test a Service

1. Data passed to the service through PRIDataRequest field in the Request Header resides in the **PRIDataRequest.xml** file. Service Developers can manipulate field values in PRIData.xml file to test the service.
2. The developer loads the default.html file in their web browser, types the URL of the service being tested in the Text field, chooses either "Get" or "Post" Submit Methods and clicks the "Go" Button. PCS then extracts the data from the PRIDataRequest.xml file and packages it into the request header's PRIDataRequest variable. A PCS Request that includes the PRIDataRequest in the Request Header is then sent to the service.
3. The Service will extract any data it needs from the PRIDataRequest Header variable and send a response. PCS loads the middle frame with this response sent by the service. This code to extract the data from the Request Header is not part of the PCS.
4. PRIDataResponse field will be extracted from the Response Header received from the Service and saved in the **PRIDataResponse.xml** file. The service developer can examine this file to check data values sent by the service being tested.
5. To see service results in different layouts, the developer selects the Size from the "Layout" drop down list and clicks the "Go" button. PCS then displays the service response in the layout they just selected. For 50:50 layout the service response will be displayed in Left side of the content frame. For 30:70 layout user gets the option to choose the side where they want the response from the service to display. For 100% layout the service response goes to the entire content area.

### 10.2.1.5 How to Install PCS

System Requirements:

- ÷! **Web Server:**  
Microsoft IIS 4.0 or above. IE 5.0 or above MUST be installed on the Web server.
- ÷! **Client Machine:**  
Microsoft Internet Explorer 4.x or 5.x or 6.x  
Netscape Communicator 4.x

The following steps will guide the developer in installing PCS on the IIS web server.

1. Unzip the "Portal Connector Stub.zip" file, which will create "Portal Connector Stub" Folder and extract the files into it.
2. Share the PCSDData folder under the 'PortalConnectorStub' folder as 'PCSDData' and set proper permissions so that the Service Developer's ID will have permissions to read/write into this folder.
3. The following steps are for IIS web servers.
  - a. Create a virtual Directory mapped on the above folder.
  - b. Set the Default Document for the above virtual directory to default.html

4. To test if the installation was successful, please use the following steps:
  - a. Launch the PCS in a browser.
  - b. Type the URL for the Test.asp in the 'Service URL' box on the left side of the 'Go' button. The Test.asp resides in the "TestService" Folder under the "Portal Connector Stub" folder.
  - c. The result of Test.asp will be displayed in the desired frame.

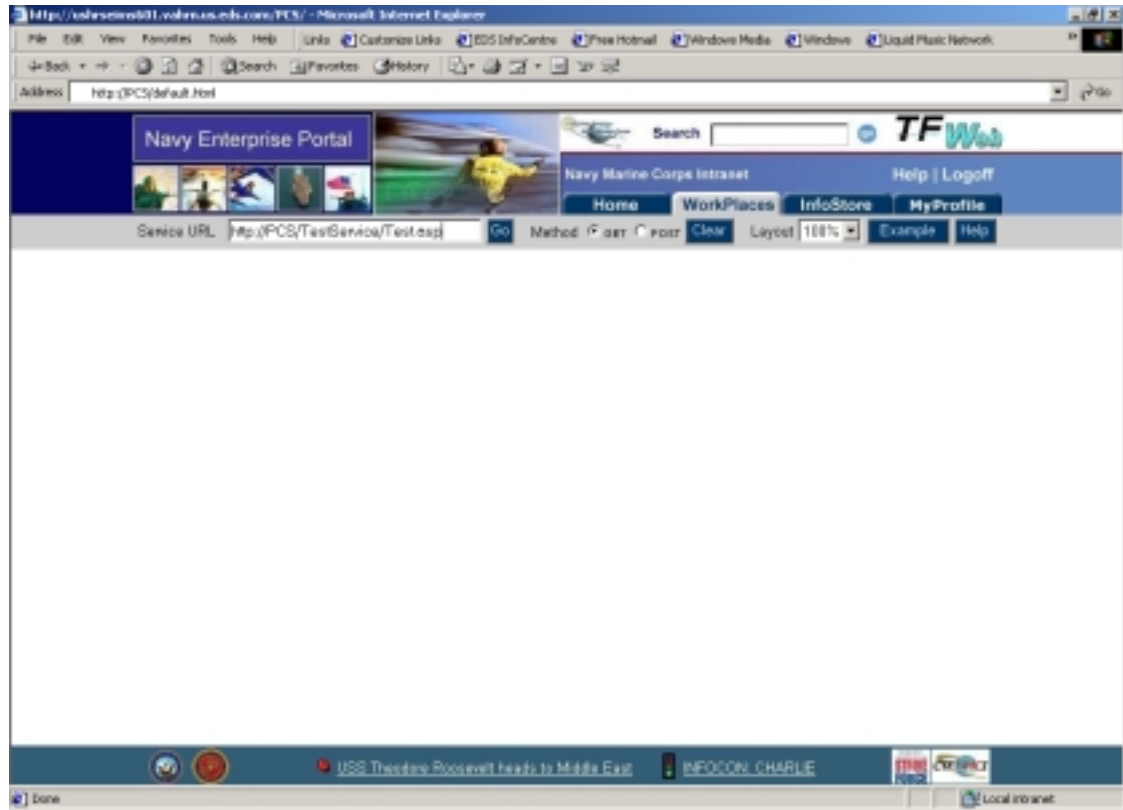


Figure 10–4: PCS

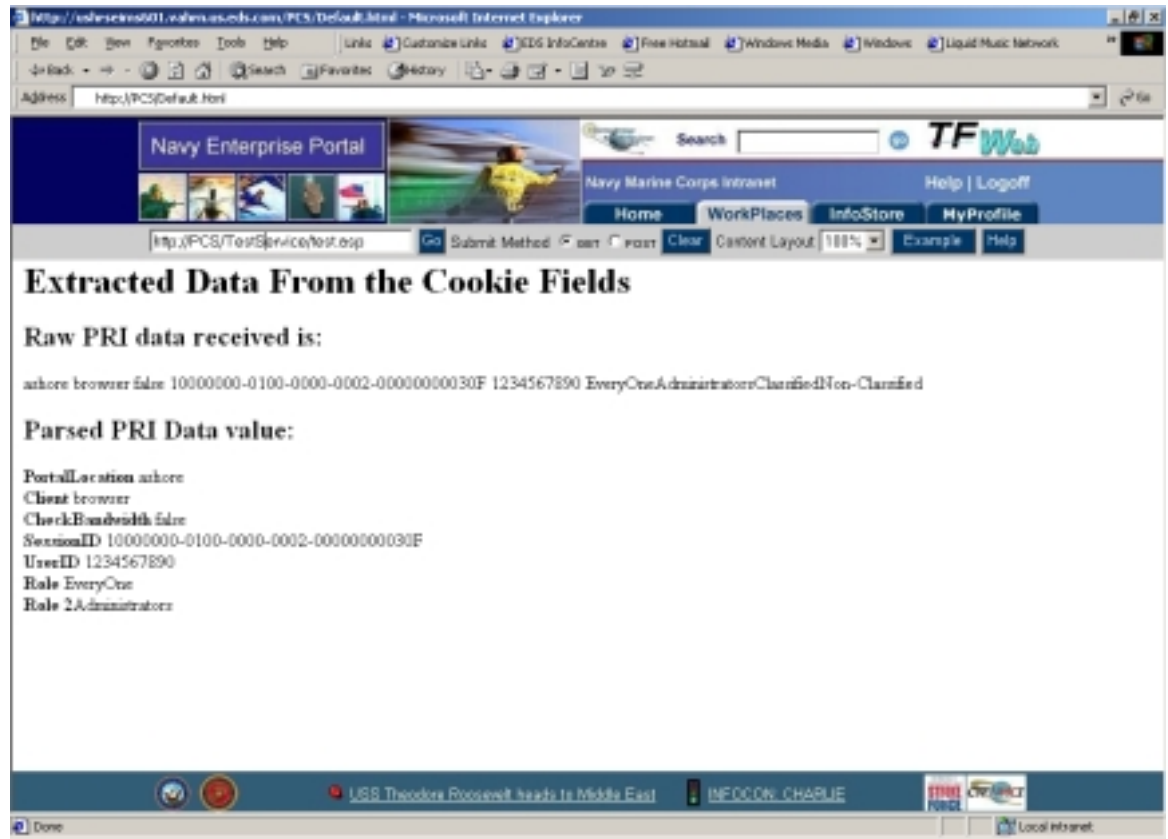


Figure 10-5: Extracted Data from the Cookie Fields

## 11 Application Owner/Analyst Guidance

The process for migrating an existing application into the Navy portal is designed to ensure that the target application meets all portal standards, security requirements, does not utilize a data environment duplicative of an existing authoritative data source, and does not provide a duplicative service.

The process begins with the service provider determining the applicability of migrating the application to the Navy portal. Next, a review of existing services and data sources is conducted to identify various duplication issues. Once the decision is made that it is appropriate to migrate the application to the Navy portal, the developer submits an IntenttoMigrate notification to the Task Force Web Application Migration Customer Support (AMCS) team. An AMCS officer will be assigned to the application who will assess the application, help to identify overlapping applications and data sources, and assist in compiling the RequesttoMigrate for submission to the Task Force Web AMCS team. After review, the AMCS team will forward applicable portions to the beta test labs for final technical review prior to integration. This chapter will discuss the AMCS processes and the specific steps required to complete each.

### 11.1 Pre-Service Registration Phase

Before starting the TFWeb integration process, the developer must answer for themselves a number of questions:

- ÷! What is my TFWeb integration goal (see Section 11.1.1)?
  - . ! What is my virtual interest group (see Section 11.1.1.1)?
- ÷! Is there an approved DoN/DoD application or service already in existence that provides this service/content (see Section 11.1.2)?
  - . ! How do I find other registered services on the Enterprise Portal (see Section 11.1.2.1)?
  - . ! How is “best of breed” determined (see Section 11.1.2.2)?
- ÷! Is my application/service already web-enabled (see Section 11.1.4)?
  - . ! If so, now what (see Section 11.1.4.1)?
  - . ! If not, should I web-enable it (see Section 11.1.4.2)?

Once these factors have been determined, the program, application, or content manager will have the data needed to determine what migration plan/POA&M will be required to achieve their targeted level of integration.

#### 11.1.1 Determining TFWeb Integration Goals

What is a “web-enabled” application? This term is often misunderstood. A “web-enabled” application is simply an application or service that is accessed within the context of a browser. This includes, but is not limited to, applications/technologies such as Java (beans, applications, scripts, applets (signed), server pages), Active Server Pages (ASPs), ActiveX components (signed), multimedia and other approved plug-ins.

Regardless of your current web-posture, there are certain key things that a developer, program, application, or content manager must determine before considering integration into the Navy Enterprise Portal:

- ÷! Determining Communities of Interest.
- ÷! Market review of existing services and content.
- ÷! Supportability and Maintainability.

#### **11.1.1.1 Determining Communities of Interest**

A key objective of web-enablement is the cross-pollination of data within and across communities of interest. WEN service providers are required to determine the virtual interest group (as shown by the taxonomy in Section 5) for their application. This determination is based upon the types of information services and/or data that are common to the community's processes or business operations and whether they would benefit from web-enabling as well as portal integration and dissemination. Combining services within virtual interest groups will illustrate size, priority, and complexity of data/information and application sharing and aid in determining cost-benefit and other intangible benefits (e.g., reduction in system operator/administrator task complexity). The managers of each community of interest (e.g., ASNRDA-CHENG/OPNAV for Battleforce information requirements) will provide the developer with the location of their authoritative data source(s) through the WEN IT Governance Board/TFWeb process. The goal is to provide an integrated data environment that will persist the appropriate data across the enterprise, and promote application re-use/consolidation.

#### **11.1.2 Reviewing Existing Services**

##### **11.1.2.1 Market Review of Existing Services and Content**

Program, application, service, or content managers should review the existing applications (commercial or otherwise) for overlapping capabilities. Build/Buy/Re-engineering decisions should be predicated on examining the list of existing services and content and their respective descriptions to ascertain whether an existing service or content can be reutilized. In short, is there an approved DoN/DoD application or service already in existence that provides this service/content?

##### **11.1.2.2 Registered Services and “Best of Breed” Determination**

Information regarding current registered services and content is found at the site of the master registry or from AMCS.. If a new service or content is being proposed for addition to the Enterprise Portal environment, an Intent to Migrate package (Section 8.2) must be submitted to the AMCS for review. AMCS verifies that there are no applications in the WEN environment that provide overlapping functionality or content, and that the implemented technologies, styling, and supportability requirements provide for TFWeb integration.

In the event that there is overlapping functionality, AMCS works with the application owners to understand and document the overlap and develop a migration plan. If a migration plan cannot be agreed upon by AMCS and the concerned application owners, the documentation is given with a recommendation to the Task Force Web Executive Steering Group, which then determines which applications will be allowed to integrate with the portal. The ESG may also make recommendations to OPNAV and other Echelon II commands regarding resolution of application/data overlap. This decision is based upon the following criteria:

- ÷! Technical/Architectural analysis performed by an independent TFWeb engineering team. This analysis includes all relevant engineering requirements (e.g., security) as defined by this and other DoD/DoN/TFWeb guidance.
- ÷! Operational Advisory Group Analysis. An OAG comprised of members from the appropriate service elements evaluates the applications for use in their environments to meet their operational needs. Several functional groups already exist and these are utilized when possible.
- ÷! Business Case analysis. Each application provider is required to build a business case analysis for evaluation. This includes review of the funding requirements, ILS Plan, and other similar documentation.

### **11.1.3 Supportability and Maintainability**

A product that is successfully integrated into the Enterprise Portal environment will be unsuccessful if it is not adequately supported. Each WEN service provider is required to show an ILS Plan (which includes all 10 elements of logistics support) that follows TFWeb standards for maintainability and supportability.

### **11.1.4 Web Enablement Determination**

Being “web-enabled” does not mean, “TFWeb-ready”. “TFWeb-ready” connotes that the developer has followed the registration, development, integration/testing, and deployment processes laid forth in this document, and been approved by the WEN IT Governance board.

#### **11.1.4.1 Existing Web-Enabled Applications**

Even though web enabling is not equivalent to being TFWeb ready, being web-enabled in some form will only help in this process. The main issues that will impact the developer with regards to TFWeb integration would be:

- ÷! Implemented web technologies (and appropriate versions) specified in the WEN Technology Baseline (e.g., Java/J2EE, Perl, CGI,)
- ÷! Presentation styling. The developer's web presence may be in conflict with TFWeb promulgated styling conventions or incompatible with the portal interface.
- ÷! Implemented naming conventions and data interoperability standards (e.g., XML).

The ultimate determination of whether or not to undertake tasking to realign or retrofit existing web-enabled applications for integration into the Enterprise Portal environment is left to the program, application, or content manager. It is strongly recommended that the entire TFWeb registration process be reviewed prior to these undertakings.

#### **11.1.4.2 Non-Web-Enabled Applications**

Before launching into an intensive integration effort to “webify” an existing Navy service or application, it must first be determined whether there is inherent value in “webifying” the particular application. It may not make sense to web enable every application or service. In many cases, the application may not be integrated into a web-based environment, but the data it provides may be hosted on the portal as relevant content. This section of the guidance document identifies a set of criteria that can be used to evaluate an application or service for “whole” or “partial” web enabling. The following should be used as general guidance for the program, application, or

content manager to determine whether or not they should endeavor to web-enable their application, and then integrate it into the Enterprise Portal environment.

The criteria identified to date include the following:

- ÷! Information Services
- ÷! Real-Time Versus Non-Real-Time
- ÷! Service/Application User environment.
- ÷! User/Administrator

It is important to note that while all applications may not require web-enabling, and therefore, do not require integration into the Enterprise Portal environment, all applications will be subject to review by the appropriate program, application, or content manager.

#### **11.1.4.2.1 Information Services**

If your application provides some content that would be usable by other elements of the Enterprise Portal, then it is a candidate for some level of TFWeb integration. It is important to remind developers that TFWeb 'web-enabling' is not equivalent to simple "web enabling." It does not necessarily mean that the application runs within the context of a browser. It may simply mean that the application offers up its data for browser-presentation rendering by the enterprise portal engine. It also means providing of content to a shared/common data environment.

Determination of content relevance across the enterprise is determined, in part, by identifying virtual interest group, and coordinating with the appropriate authoritative data source.

#### **11.1.4.2.2 Real-Time Versus Non-Real-Time**

The Web or Internet is not a real-time medium. There is no intention of firing a weapon from a web browser. Real-time, rapid response systems are not good candidates for web enabling. However, having said that, there may be status information from a real-time system that can and should be web enabled and made available.

#### **11.1.4.2.3 Service/Application User Environment**

Web applications are by definition multi-tiered network architected services that deliver content (e.g., application components or data) based on an established network, data persistence, and security model. There are application and service environments that are fundamental to operational requirements (e.g., small community of interest users that are distributed across large areas) regardless of user community size. For example, there's a community of senior flag officers that are extremely essential to operational requirements as a community of interest. These users require specific applications/data, unique to their environment, with high levels of security, which must operate in a distributed manner.

#### **11.1.4.2.4 User/Administrator**

Much of the development effort of any application goes into the management interface. While required, this interface may be used by a small fraction of the total number of users. It is recommended that application owners focus first on the end-user interface to deliver as much capability to the end user as resources and time permits. Rewriting existing management interfaces often has a cost higher than any benefit that will result to the managers. New applications, however, should expect that all functionality is web-based when originally developed.

### **11.2 Intent to Migrate**

Once the decision is made to migrate an application to the Navy Enterprise Portal, the developer must notify the TFWeb team of the intent to migrate an application or service. Completion of these steps (Section 8.2) is required for application-specific AMCS/AMTS support. This serves to notify all concerned of integrator or sponsor's intent to provide a given service via the TFW portal



and allows migration tracking and preparation for receipt of required information for migration. It also helps to prioritize and focus technical support assets based on the impact of the application, timeframe of migration, and difficulty of transition. The following actions are taken by the AMCS contact assigned based on the application owner submission.

#### **11.2.1 Submission to the application information database.**

This database is maintained by AMCS. AMCS will review the submission to determine if fields are complete and understandable. Descriptions should be useful and thorough? Yes/no answers may need further comment other information may be required to be tracked for the application. Implementation dates should be reasonably achievable. The AMCS contact for the application ensures the submission is properly reviewed and entered in the database and notifies the Echelon II contact of any changes made during the review process to their submission.

#### **11.2.2 Integration Level Appropriateness**

**Level 1 applications require specific detailed explanations why it cannot be Category 2 and require AMCS OIC's authorization for integration into the portal.** Category 2 applications may be approved by the TFW Echelon II liaison for preexisting applications and for applications that require immediate rollout beyond the portal user base.

#### **11.2.3 Identify if the program uses Java, JavaScript, ActiveX, or plugins**

Additional review/analysis of these issues are coordinated through the TFW Echelon II liaison with the appropriate AMCS department head covering technical issues. In addition, non-mobile ActiveX or plugins must have a satisfactory distribution plan in compliance with applicable NMCI and IT-21 policies.

#### **11.2.4 Examine the application database for similar programs that are currently under development**

The AMCS contact reviews data sources for possible data overlap and examines overlapping applications reported by other application owners. If possible overlap exists, they interface with program managers of all concerned programs and data sources to determine exact functionality, user base, and IT requirements. If consolidation possibilities exist, they brief AMCS OIC on overlap and application/data owners' intentions to determine any further action warranted.

#### **11.2.5 Determine current security model and whether IATO/ATO exists, or is required**

Determine what changes, if any, are required to the current application security model in order to integrate application. Is the current security model compatible with TFW security model (issues like SSN)? If no changes are required, the AMCS contact ensures a copy of existing IATO/ATO cover sheet is sent to AMCS IA.

#### **11.2.6 Determine XML integration requirements**

Evaluate the plan for design and registration of the schema and other XML documentation? Does this need to be coordinated with other commands using similar data? Ensure that the application owner is familiar with the DoN XML instruction.

### **11.3 Service Registration**

The Service Registration package (Section 8.2) is submitted to the AMCS contact after development has been completed. AMCS performs the following items as part of the package review. This section also applies to changes required as part of the beta testing procedures prior to restarting testing.

### **11.3.1 Verify completeness and accuracy of portal metadata**

This should include the directory entry text, category, description, application owner, and application "customer service" contact. This is information available to any user of the portal. Is it sufficient to determine whether access to an application is required and how to obtain access? Does it address intended user base and purpose of the program? Similar programs directed at other userbases should be mentioned in the description.

### **11.3.2 Verify migration plan for level of integration is submitted**

Retain copy of migration plan in AMCS Echelon II notes for future reference. Brief migration plan to AMCS OIC for approval.

### **11.3.3 Ensure IATO/ATO has been updated if security model changed for TFW migration**

Provide copy of IATO/ATO cover letter to AMCS IA for reference documentation. Submission of full accreditation paperwork is not required unless determined necessary by AMCS IA.

### **11.3.4 Verify initial access control list submitted along with information describing method of updating ACL**

Verify method is compatible with current portal capabilities and user expectations. If access cannot be given in a timely manner, ensure it is indicated in the application description visible to the user. Review and approve appropriate roles for application visibility.

### **11.3.5 Portal Compliance Testing**

AMCS liaison shall be provided a temporary login with access to key features of the application. In the event access to key areas cannot be provided due to security/access issues, a representative of the application developer will be made available for the demonstration. Spot check to ensure claimed capabilities of user description are provided and significant limitations are documented. Spot check HTML used is "portal compliant" (no frames). Record all concerns, discuss with developer or program manager. Submit any unresolved discrepancies to AMTS (for pre-beta review) or as part of the AMCS beta testing notes. This is not intended to be a thorough review of the program. Rather, it serves to ensure any obvious issues are recognized and documented prior to the beta testing process to help expedite testing

### **11.3.6 Review summary of testing accomplished**

Summary should include duration, type of users, type of test scripts performed, type of data used, and environment in relation to the production platform. What is the risk that the program will fail beta testing? Has there been sufficient operator testing to ensure utility in the production environment?

### **11.3.7 Review portal integration information submitted**

To ensure effective use of testing time and to allow maximum preparation time for testing, all required portal integration information should be submitted as part of the request. While further changes may be necessary or desirable, this allows for a package of all required information to be submitted from AMCS to the beta test site. Ensure integration module code has been provided

Identify if substantive revisions have been made to sample code. Module code should be fully documented and readable. Evaluate code for posting to open source site (based on differences from baseline code). If review of code is required, submit request to AMTS. Are there any reusable components that should be separately maintained?

### **11.3.8 DoN XML guideline compliance**

If the application does not meet DoN XML guidelines, ensure migration plan is submitted and approved by AMCS OIC.

### **11.3.9 Set next review date**

General guideline is 1 year if all requirements met or halfway to next integration level (3 month minimum) if a migration plan has been submitted for non-Level 3 integration or noncompliant XML are used. If IATO has been submitted, review date should be prior to its expiration. By this review date, a member of AMCS will review documentation and implementation history, and determine if any additional information is required. Milestones in migration plans or further functionality development will be reviewed. Also, database information will be verified.

### **11.3.10 Verify database entry is complete and accurate in AMCS application database**

#### **11.3.11 Technical Review**

AMCS may request a technical review at any time from the AMTS or alternate source. This technical review may evaluate code base, technology, mobile code, or security among possible areas. In some cases this is used to evaluate leading-edge technology and possible unforeseen impacts on TFW environment. In other cases, it is used to check for compliance with TFW architecture in a more thorough fashion than is possible in the beta testing environment. If necessary to evaluate an application's readiness for migration, this review is completed and any discrepancies resolved prior to permission for beta testing. The AMCS is the final arbiter of whom discrepancies are required to be resolved prior to beta testing, though review of an AMCS decision may be requested from the TFW Executive Steering Group.

#### **11.3.12 Configuration Verification**

Verify configuration of any local application servers or local remote module servers are documented. The ability of local infrastructure to support numbers of users intended should be documented as well as ability to scale to additional users. Any known scalability issues should be documented.

#### **11.3.13 Ensure application is logged in the DON CIO Data Management and Interoperability Repository.**

DMIR is currently in the beta testing stage. This is an optional requirement until full functionality in 2002.

#### **11.3.14 Verify all application data structures and data interfaces are documented.**

Databases should be accessible independently of application if underlying database engine and security supports. Data interfaces should also be accessible independently.

#### **11.3.15 Verify AMCS OIC has approved migration plan for application/data overlap.**

Migration plan should address duplicative applications and data sources and their planned resolution. Migration plan is not due until final review of application after beta testing.

#### **11.3.16 Ensure developer requirements for future capability upgrades of theWEN architecture are documented if current implementation does not allow for desired developer functionality.**

This should consist solely of architecture or cross-application services, not those useful only to a single application. This helps prioritize additional requirements based on the ability of the developer community to capitalize upon the new features.

## 11.4 Application/Service Delivery Phase

### 11.4.1 Application Acceptance

Only applications that have completed the migration request process with TFWeb are submitted (e.g., application components, links/icons, datafill, DTDs/Schemas) for integration in Enterprise Portal. All applications need to ensure compliance with required DoN/DoD policies/ infrastructure release for their virtual interest group (e.g., DII-COE) that the portal is running. Any waivers or deviations should be annotated (with approval) within the application release notification message. The TFWeb process does *not* supercede individual program, application, or content manager processes. It is expected that the Enterprise Portal will receive applications that have gone through internal system engineering and logistics processes (e.g., CCB, internal testing, CM).

### 11.4.2 Application Delivery

Each application is expected to deliver system and administration documentation that conforms to Enterprise Portal documentation guidelines (e.g., XML). This includes software operation and concise loading instructions to enable users/administrators to load and administer the applications with minimum intervention. The instructions should also include load verification and load back-out procedures.

Once the application is successfully loaded into the TFWeb developmental portal environment, the developer will then continue with the remainder of the self-certification procedure, moving into the performance criteria.

### 11.4.3 Application Integration

The developer is encouraged to notify TFWeb of an impending application release no later than 30 days prior to portal integration. This gives the TFWeb team time to arbitrate schedule conflicts with other application developers.

The application integration process differs depending upon the type of application to be integrated, and the level of integration the application is achieving. In all cases the goal is to provide the developer a process and supporting infrastructure by which they can develop, test, and certify their application(s) for use in the Enterprise portal environment with a minimum involvement by a core TFWeb team or other external agencies.

This section is highly dependent on the final Enterprise portal architecture and specific applications selected as well as the current web enabling status of existing Navy services.

#### 11.4.3.1 Developer Integration Environment

It is planned that prior to integration with the run-time Enterprise portal, application developers who are attempting any level of integration above hotlink integration will be required to do some level of integration and testing with a functional portal baseline. The developmental portal environment(s) will be accessible from any security enclave. Developers will be able to schedule, integrate, test, and manage their applications wholly within the portal environment without involvement of any TFWeb personnel (unless specific arbitration or problem resolution services are required). Access will be restricted to registered developers.

## Appendix A: Department of Defense and Department of the Navy Authority References

There are many DoD and Navy policies and guidance that need to be followed to provide consistency across applications. Sometimes, the timing and overlap between these policies and guidance can cause the local commands confusion over which apply. The following list includes some of the more relevant policies and offices for use as references in preparing for Enterprise portal integration.

- ÷! ASD(C3I) Memorandum "Accessibility of DoD Web Site to People with Disabilities" dated 21 Jul 2000
- ÷! Section 508 Of The Rehabilitation Act As Amended, 29 U.S.C. Section 794d, Requires That When Federal Agencies Develop, Procure, Maintain Or Use Electronic And Information Technology (IT).
- ÷! ASD(C3I) Memorandum "Policy Guidance for Mobile Code Technologies in Department of Defense (DoD) Information Systems," dated 7 November 2000
- ÷! CNO WASHINGTON DC/N6// Policy Update: Use Of Portable Electronic Devices In The Navy
- ÷! FLTINFOWARCEN NORFOLK VA 131301Z JUL 00
- ÷! DIRNSA FT GEORGE G MEADE MD 171857Z JAN 01
- ÷! SECNAVINST 5510.36 DON Information Security Program Regulation.
- ÷! OPNAVINST 5239.1 Navy Information Assurance Program
- ÷! SECNAVINST 5720.47 Department of the Navy Policy for Content of Publicly Assessable World Wide Web Sites
- ÷! NAVCIRT Advisory 00-28 "Personal Digital Assistants Security Considerations"
- ÷! DIRNSA Information Assurance Advisory No. IAA-001-01 "Personal Electronic Device Security Guidance"
- ÷! ASD(C3I) Memorandum "Increasing The Security Posture Of The NIPRNet" dated August 22, 1999
- ÷! DoD Directive 5200.28 "Security Requirements for Automated Information Systems"
- ÷! DoD Instruction 5200.40 "DoD Information Technology (IT) Security Certification And Accreditation Process (DITSCAP)"
- ÷! CJCS Memo CM-510-99 Information Operations Condition dated 10 Mar 1999
- ÷! CNO WASHINGTON DC 181840Z MAY 99 NAVY INFORMATION OPERATIONS CONDITION (INFOCON) IMPLEMENTATION
- ÷! Fleet Internet Security Handbook (FISH) Advisories & Bulletins: at <http://infosec.navy.mil>

- ÷! CNO WASHINGTON DC 211137Z AUG 00 NAVY-MARINE CORPS FIREWALL POLICY  
NAVY-MARINE CORPS NIPRNET FIREWALL CONFIGURATION BASELINE DATED: 01  
FEB 01 Available At SIPRNET URL - <http://Infosec.Navy.Smil.Mil/Fleet/>
- ÷! CNO WASHINGTON DC 091934Z JUN 00 NAVY CERTIFICATION & ACCREDITATION  
OF SYSTEMS AND NETWORKS
- ÷! SPAWAR PMW-161 Naval VPN Product Requirements
- ÷! OPNAVINST 2201.2 "CNO-CMC Navy and Marine Corps Computer Network Incident  
Reporting Policy "
- ÷! DON Policy For Content Of Publicly Accessible World Wide Web Sites
- ÷! DoD Policy Memorandum "Web Site Administration" dated Dec 7 1998
- ÷! SECNAVINST 5720.44A, "Department of the Navy Public Affairs Policy and Regulations"
- ÷! SECNAVINST 5430.97, "Assignment of Public Affairs Responsibilities in the Department  
of the Navy"
- ÷! SECNAVINST 5211 .5D, "Department of the Navy Privacy Act (PA) Program"
- ÷! SECNAVINST 5720.42F, "Department of the Navy Freedom of Information Act (FOIA)  
Program"
- ÷! SECNAVINST 5239.3 "Department of the Navy Information Security (INFOSEC)  
Program"
- ÷! DoD Directive 5040.5 "Alteration of Official DoD Imagery"
- ÷! DoD Joint Technical Architecture (JTA) Version 4.0 dated 2 April 2001 located @  
<http://www-jta.itsi.disa.mil/>
- ÷! DoD Policy Guidance for use of Mobile Code Technologies in the Department of Defense  
Information Systems, dated November 7, 2000
- ÷! DOD Instruction 5230.29 "Security & Policy Review of DOD Information for Public  
Release"
- ÷! NAWCADINST 5728.1 Policy for Release of Information on WWW servers
- ÷! DON Interim XML Policy. "INTERIM POLICY ON THE USE OF EXTENSIBLE MARKUP  
LANGUAGE (XML) FOR DATA EXCHANGE of 6 Sept 2001"  
[http://quickplace.hq.navy.mil/QuickPlace/navyxml/Main.nsf/h\\_081AA1B352C96B1A85256ACB006618FF/DAF10483C68509CB85256ACB006645C3?OpenDocument](http://quickplace.hq.navy.mil/QuickPlace/navyxml/Main.nsf/h_081AA1B352C96B1A85256ACB006618FF/DAF10483C68509CB85256ACB006645C3?OpenDocument)
- ÷! Initial DONXML Developer's Guide - 29 October 2001 - Department of the Navy, Office  
of the Chief Information Officer, XML Work Group  
[http://quickplace.hq.navy.mil/QuickPlace/navyxml/Main.nsf/h\\_Toc/7E0EB98C7FBA267D85256AF5006C8B3D?OpenDocument](http://quickplace.hq.navy.mil/QuickPlace/navyxml/Main.nsf/h_Toc/7E0EB98C7FBA267D85256AF5006C8B3D?OpenDocument)
- ÷! A collection of applicable DoD Webmasters Policies And Guidelines can be found at  
<http://www.defenselink.mil/webmasters/>

## Appendix B: Service Code Samples/Templates

### Hyperlink Integration

#### JSP Repository Example

```
<HTML>

<HEAD>
<TITLE>Integration 1</TITLE>
</HEAD>
<BODY>
<P>Sample Level 1 (Hyperlink) Integration Service Module</P>

<!--
// ***** MODIFY THE FOLLOWING LINE *****//
-->

<a href="http://ushrseims312.vahrn.us.eds.com/servlet/RRTest/DebugApplication.jsp"
target="_blank">Click here</a>

<!--
// *** Change the http://.... URL to indicate the absolute path for ***//
// *** the server where the test application is actually installed the ***//
// *****//
-->

</BODY>
</HTML>
```

#### JSP Application Example

```
<%/*****
* Title: DebugApplication.jsp
*
* Description: Server page that prints out debugging information.
* Copyright: Copyright (c) 2001
* Company: EDS
* @author TFW Application/Repository Migration Team
* @version 1.0
*
* THIS IS SOURCE CODE PUBLISHED FOR DEMONSTRATION PURPOSES
*
* This page reads all the query parameters and echos them back in HTML.
*
*/%>

<%@ page language="java" %>
<%@ page import="
    javax.servlet.*,
    javax.servlet.http.*" %>
```



```
<HTML>
<HEAD>
<TITLE>Debugging Application</TITLE>
</HEAD>
<BODY>
<H1>Application Successfully Accessed</H1>
<%

Enumeration parms = request.getParameterNames();

if (parms.hasMoreElements()) {
    out.println("<H3>Parameters passed in:</H3>");
}

while (parms.hasMoreElements()) {
    String pName = (String) parms.nextElement();
    String[] sa = request.getParameterValues(pName);
    out.println("<B>"+pName+"</B>&nbsp;&nbsp;&nbsp;");
    for (int i=0; i<sa.length; i++) {
        if (i>0) out.print(", ");
        out.println(sa[i]);
    }
    out.println("<BR>");
}

%>

</BODY>
```

### ASP Repository Example

```
<%    response.redirect "http://swat.nosc.mil/tfw/app.asp"%>
<html>
<head>
    <title>test</title>
</head>
    <body>

    </body>
</html>
```

### ASP Application Example

```
<html>
<head>
<title>test</title>
</head>

<body>
```

Hello World from TFW .asp page!

</body>

</html>

## XML Schema and XSL Style Sheet

For all three examples described above, the database, XML schema and XSL style sheet were all the same. The following XML schema describes the structure and semantics of the XML file to be generated by the example scripts. In the future, a standard schema will be supplied to aid in future data aggregation. The XML schema is defined as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="Amphibious">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="Auxiliary">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="FixedWing">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="Lafayette">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="LosAngeles">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="MineWarfare">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="NavyCombat">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Amphibious"/>
        <xsd:element ref="Auxiliary"/>
        <xsd:element ref="FixedWing"/>
        <xsd:element ref="MineWarfare"/>
        <xsd:element ref="NavySatellite"/>
        <xsd:element ref="RotorWing"/>
        <xsd:element ref="Submarines"/>
        <xsd:element ref="SurfaceCombatants"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="NavySatellite">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="Ohio">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="Osprey">
```

```
<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="vehicleInfo"/>
  </xsd:sequence>
  <xsd:attribute name="craft_type" type="xsd:string" use="required"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="RotorWing">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="SeaCobra"/>
      <xsd:element ref="SeaKing"/>
      <xsd:element ref="SeaKnight"/>
      <xsd:element ref="SeaStallion"/>
      <xsd:element ref="SeaHawk"/>
      <xsd:element ref="Seasprite"/>
      <xsd:element ref="Osprey"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="SeaCobra">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="vehicleInfo" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="craft_type" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="SeaHawk">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="vehicleInfo"/>
    </xsd:sequence>
    <xsd:attribute name="craft_type" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="SeaKing">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="vehicleInfo"/>
    </xsd:sequence>
    <xsd:attribute name="craft_type" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="SeaKnight">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="vehicleInfo"/>
    </xsd:sequence>
    <xsd:attribute name="craft_type" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="SeaStallion">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="vehicleInfo"/>
```

```

        </xsd:sequence>
        <xsd:attribute name="craft_type" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="Seasprite">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="vehicleInfo"/>
        </xsd:sequence>
        <xsd:attribute name="craft_type" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="Seawolf">
    <xsd:complexType/>
</xsd:element>
<xsd:element name="Sturgeon">
    <xsd:complexType/>
</xsd:element>
<xsd:element name="Submarines">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Lafayette"/>
            <xsd:element ref="LosAngeles"/>
            <xsd:element ref="Ohio"/>
            <xsd:element ref="Seawolf"/>
            <xsd:element ref="Sturgeon"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="SurfaceCombatants">
    <xsd:complexType/>
</xsd:element>
<xsd:element name="vehicleInfo">
    <xsd:complexType>
        <xsd:attribute name="pilot" type="xsd:string" use="required"/>
        <xsd:attribute name="inRepair" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="false"/>
                    <xsd:enumeration value="true"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="call_sign" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="ADW91"/>
                    <xsd:enumeration value="ZFW75"/>
                    <xsd:enumeration value="ZZZZZ"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
</xsd:element>
</xsd:schema>

```

The XSL style sheet is defined as follows:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output method="html" indent="yes" />

<xsl:template match="/">
<html>
<head>
<title>TFW response UI screen for display of Navy Assets</title>
</head>
<body>
<div align="center" >
<b><h2>
  <p>Task Force Web (TFW)</p>
  <p>Navy Asset View -- RESPONSE PAGE</p>
</h2>
</b>
</div>
<center>
<table border="1">
  <tr>
    <th>VEHICLE TYPE</th>
    <th>VEHICLE NAME</th>
    <th>CRAFT TYPE</th>
    <th>PILOT</th>
    <th>IN REPAIR</th>
    <th>CALL SIGN</th>
  </tr>
  <xsl:for-each select="/NavyCombat/*">
    <xsl:variable name="vehtype" select="name(.)" />
    <xsl:choose>
      <xsl:when test="$vehtype='RotorWing'">
        <xsl:for-each select="/NavyCombat/RotorWing/*">
          <xsl:variable name="subname" select="name(.)" />
          <xsl:variable name="crafttype" select="@craft_type" />
          <xsl:for-each select="*">
            <tr>
              <td><xsl:value-of select="$vehtype" /></td>
              <td><xsl:value-of select="$subname" /></td>
              <td><xsl:value-of select="$crafttype" /></td>
              <td><xsl:value-of select="@pilot" /></td>
              <td><xsl:value-of select="@inRepair" /></td>
              <td><xsl:value-of select="@call_sign" /></td>
            </tr>
          </xsl:for-each>
        </xsl:for-each>
      </xsl:when>
      <xsl:when test="$vehtype='Submarines'">
        <xsl:for-each select="/NavyCombat/Submarines/*">
          <xsl:variable name="subname" select="name(.)" />
          <tr>
            <td><xsl:value-of select="$vehtype" /></td>
            <td><xsl:value-of select="$subname" /></td>
            <td>N/A</td>
            <td>N/A</td>
            <td>N/A</td>
            <td>N/A</td>
          </tr>
        </xsl:for-each>
      </xsl:when>
    </xsl:choose>
  </xsl:for-each>
</table>
</center>
</div>
</body>
</html>
</xsl:template>
```

```

        <td>N/A</td>
      </tr>
    </xsl:for-each>
  </xsl:when>
  <xsl:otherwise>
    <tr>
      <td><xsl:value-of select="$vehtype" /></td>
      <td>N/A</td>
      <td>N/A</td>
      <td>N/A</td>
      <td>N/A</td>
      <td>N/A</td>
    </tr>
  </xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</table>
</center>
</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

Navy web enablement is the implementation of interoperable web technologies across the Naval infrastructure allowing subscribers and publishers (users and providers) of content to pull or push services as required to perform operational or business transactions. A Navy web transaction is the execution of a web-service. The service centric access for the Navy is depicted in Figure 6-1.

## Application/Data Integration

### Java Server Page (JSP) Example

```

<%/*****
* Title: SampleService.jsp
*
* Description: Server page that takes service request from portal connector.
* Copyright: Copyright (c) 2001
* Company: EDS
* @author TFW Application/Repository Migration Team
* @version 1.0
*
* THIS IS SOURCE CODE PUBLISHED FOR DEMONSTRATION PURPOSES
*
* This page is deployed on the TFW Service Repository application server.

```

- \* It communicates with the TFW Portal via HTTP/HTTPS calls from portal to repository.
- \* It accomplishes 5 tasks.
  - \* 1. Receive and parse PRIData request in HTTP header
  - \* 2. Make a HTTP or SOAP call to retrieve data
  - \* 3. Generate the response header using the results of step 2
  - \* 4. Create a PRIData response and insert in the returned HTTP header
  - \* 5. Send HTTP or SOAP call results back in HTTP body
- \* /%>

```

<%@ page language="java" %>
<%@ page import="
    java.io.*,
    java.net.*,
    javax.servlet.*,
    javax.servlet.http.*" %>
<%@ page import="TFWebPRI.*" %>

<%
// ===== //
// STEP 1:
// Receive and parse PRIData request in HTTP header
// ===== //

String xmlHeader = null;
xmlHeader = request.getHeader("PRIDataRequest");
if (xmlHeader == null) {
    // Should generate 403 error instead. do this for internal testing
    // xmlHeader = "<?xml version='1.0'?'><PRI_Request
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation="PRI_Request.xsd\">      <ChannelContext>
    <PortalLocation>ashore</PortalLocation>              <Client>browser</Client>
    <CheckBandwidth>true</CheckBandwidth>                </ChannelContext>
    <SessionContext>                                     <SessionID>10000000-0100-0000-0002-
000000000030F</SessionID>                               <UserID>1234567890</UserID>      <Roles>
    <Role>EveryoneHaveFun</Role>                          <Role>Comdesron
Three</Role>      </Roles>                               </SessionContext></PRI_Request\";

    response.sendError(403, "Missing or invalid PRIDataRequest");
    return;
}

```



```
};

xmlHeader = URLDecoder.decode(xmlHeader);

PRIRequest requestFromPortal;
try {
    requestFromPortal = new PRIRequest(xmlHeader);
}
catch (Exception e) {
    response.sendError(403, "Missing or invalid PRIDataRequest");
    return;
}

// ===== //
// STEP 2:
// Make a HTTP or SOAP call to retrieve data
// (for this example, open http connection to debug app)
// ===== //

// ***** MODIFY THE FOLLOWING LINE *****//

URL backendURL = new
URL("http://ushrseims312.vahrn.us.eds.com/servlet/RRTest/DebugApplication.jsp");

// *** Change the http://... URL to indicate the absolute path for ***//
// *** the server where the test application is actually installed the ***//
// *****//

URLConnection backEndConnection =
    new weblogic.net.http.HttpURLConnection(backendURL);
backEndConnection.setDoOutput(true);
backEndConnection.setDoInput(true);
backEndConnection.setRequestMethod("POST");

StringBuffer postData = new StringBuffer();
postData.append ("PortalLocation="+requestFromPortal.getPortalLocation());
```

```
postData.append("&Client="+requestFromPortal.getClient());
postData.append("&CheckBandwidth="+requestFromPortal.getCheckBandwidth());
Vector r = requestFromPortal.getRoles();
for (int i=0; i<r.size(); i++) {
    postData.append("&Role=" + r.elementAt(i));
}

backEndConnection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
backEndConnection.setRequestProperty("Content-Length", ""+postData.length());
backEndConnection.connect();

PrintStream postDataStream = new PrintStream( backEndConnection.getOutputStream() );
postDataStream.print( postData.toString() );
postDataStream.close();

// ===== //
// STEP 3:
// Generate the response header using the results of step 2
// ===== //

int rc = backEndConnection.getResponseCode();
if (rc<200 || rc>299) {
    response.sendError(rc, backEndConnection.getResponseMessage());
    return;
}

// copy the header fields from the destination to my response over
for (int i=2; ; i++) {
    String h = backEndConnection.getHeaderField(i);
    if (h == null) break;
    String k = backEndConnection.getHeaderFieldKey(i);
    response.addHeader(h,k);
}
```

```
// ===== //
// STEP 4:
// Create a PRIData response and insert in the returned HTTP
//   after the other header fields
// ===== //

PRIResponse responseToPortal = new PRIResponse(3, "FALSE", 10);
response.addHeader("PRIDataResponse", responseToPortal.extractToData());


// ===== //
// STEP 5:
// Send HTTP or SOAP call results back in HTTP body
// ===== //

BufferedReader inr = new BufferedReader(
    new InputStreamReader(backEndConnection.getInputStream())
);

String line;
while((line = inr.readLine()) != null) {
    out.println(line);
}

backEndConnection.disconnect();

%>
```

## Active Server Page (ASP) Example

Example will be available at a later date.

## Common Gateway Interface (CGI) Example in Perl

Example will be available at a later date

## Appendix C: Standards

### Technology

| Technology                  | Version |
|-----------------------------|---------|
| J2EE                        | 1.3     |
| JSP                         | 1.1     |
| BEA WebLogic                | 6.1     |
| Microsoft IIS               | 5.0     |
| EJB                         | 2.0     |
| Servlet                     | 2.2     |
| J2EE Connector Architecture | 1.0     |
| HTML                        | 4.0     |
| XML                         | 1.0     |
|                             |         |

### DoD Section 508 Compliance

The purpose of this section is to provide information on the background and issues related to implementation of Section 508 of the Rehabilitation Act and the impact of this Act on development of the TFWeb portal. Section 508 of the Rehabilitation Act amendments requires that when Federal agencies develop, procure, maintain, or use electronic and information technology, they shall ensure that the electronic and information technology allows Federal employees and the public with disabilities to have access to and use of information and data that is comparable to the access to and use of information and data by individuals without disabilities, unless an undue burden would be imposed on the agency.

### Web page accessibility checklist

**This Checklist should serve as a tool for evaluating the extent to which TFW web pages are accessible to people with disabilities.**

| No. | Questions  | Y | N | N/A |
|-----|--|---|---|-----|
| 1   | For all images, is alternative text provided?<br><br>Note: This includes images used as spacers, bullets in lists, and links |   |   |     |
| 2   | For all applets, are alternative text and content provided?  |   |   |     |
| 3   | For all image map links, is alternative text provided?   |   |   |     |
| 4   | If server-side image maps were used, are text links provided for each hotspot in the image map?                              |   |   |     |

| No. | Questions   | Y | N | N/A |
|-----|---|---|---|-----|
| 5   | For all graphical buttons, is alternative text provided?  |   |   |     |
| 6   | Is there an absence of ASCII art, and, instead, are images and alternative text used?<br>e.g., use "smile" or an image with alt text instead of: :)   |   |   |     |
| 7   | If OBJECT was used to incorporate an image, applet, or script into a page, is the information also conveyed in an alternative means in cases where the OBJECT cannot be perceived, such as with "title" or within the body of the OBJECT element? |   |   |     |
| 8   | Are long descriptions provided of all graphics that convey important information?<br>To do so: use "longdesc."  |   |   |     |
| 9   | Until most browsers support "longdesc," also use a d-link (description link) or invisible d-link.   |   |   |     |
| 10  | For stand-alone audio files, are textual transcripts of all words spoken or sung as well as all significant sounds provided?  |   |   |     |
| 11  | For audio associated with video, are captions -- textual transcripts of dialog and sounds -- synchronized with the video?   |   |   |     |
| 12  | Where sounds are played automatically, are visual notification and transcripts provided?  |   |   |     |
| 13  | For short animations such as animated "gifs" images, are alternative text and a long description provided, if needed?   |   |   |     |
| 14  | For movies, are auditory descriptions provided and synchronized with the original audio?  |   |   |     |
| 15  | If color is used to convey information, is the information also clear from the markup and/or text?<br>Hint: One way of testing this is to ask yourself whether the information is available if one is viewing it on a black and white screen.     |   |   |     |
| 16  | Are foreground and background color combinations used that provide sufficient contrast when viewed by someone with color blindness or when viewed on a black and white screen?  |   |   |     |
| 17  | For auto-refreshing or timed response pages, is a second copy of the page provided where refresh only happens after a link has been selected (until user agents provide this ability themselves)?   |   |   |     |
| 18  | Is the Web page free from any blinking or updating of the screen that causes flicker?   |   |   |     |
| 19  | Is a fallback page provided for pages that contain frames?  |   |   |     |
| 20  | For scripts that present critical information or functions, is an alternative, equivalent presentation or mechanism provided?   |   |   |     |

| No. | Questions  | Y | N | N/A |
|-----|--|---|---|-----|
| 21  | For pages that use style sheets, are the contents of each page ordered and structured so that they read appropriately without the style sheet? |   |   |     |
| 22  | If frames are used, are titles provided so that users can keep track of frames by name?  |   |   |     |

## Appendix D: Development References

TFW WEN Architecture Version 1.1.doc

TFW Portal to Repository Interface Specification V 1.0

BEA Documentation can be found at the following web site: <http://e-docs.bea.com/wls/docs61/index.html>. Documents from the BEA site that will help with the development and deployment of services on BEA site are:

- ÷! Programming WebLogic Enterprise JavaBeans at: <http://e-docs.bea.com/wls/docs61/ejb/index.html>
- ÷! Programming WebLogic JSP at: <http://e-docs.bea.com/wls/docs61/jsp/index.html>
- ÷! Assembling and Configuring Web Applications at: <http://e-docs.bea.com/wls/docs61/webapp/index.html>

For Java Coding standards please follow the standards listed at web site:

<http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>

For IIS reference at web site: <http://www.microsoft.com/>

For Jasmine reference at web site: [http://ca.com/products/jasmine/app\\_server.htm](http://ca.com/products/jasmine/app_server.htm)

For Cleartrust reference at web site: <http://www.rsasecurity.com/products/cleartrust/index.html>



## Appendix E: Terminology Glossary

| Term                         | Definition   |
|------------------------------|--|
| 3DES                         | Triple Data Encryption Standard  |
| ACID                         | Atomicity, Consistency, Isolation, and Durability  |
| ACL                          | Access Control Lists   |
| ACO                          | Administrative Contracting Officer   |
| ACS                          | Assistant Chief of Staff   |
| ACT                          | Action Collaboration Team  |
| AD                           | Active Directory   |
| ADPM                         | Architecture Development Process Model   |
| AFB                          | Air Force Base   |
| AMCS                         | Application Migration Customer Support   |
| AMTS                         | Application Migration Technical Support  |
| AoA                          | Analysis of Alternatives   |
| AOR                          | Assumption of Responsibility   |
| API                          | Application Program Interface  |
| Application/Data Integration | Involves a more closely coupled integration of an application with the Enterprise Portal System. |
| ASD                          | Assistant Secretary of Defense   |
| ASN                          | Assistant Secretary of the Navy  |
| ASP                          | Active Server Pages  |
| ATM                          | Asynchronous Transfer Mode   |
| ATO                          | Authority to Operate   |
| BAN                          | Base Area Network  |
| BFM                          | Business Financial Manager   |
| BLII                         | Base Level Information Infrastructure  |
| BPR                          | Business Process Reengineering   |
| BUMED                        | Bureau of Medicine and Surgery   |
| BUPERS                       | Bureau of Naval Personnel  |
| BWM                          | Bandwidth Manager  |
| C&A                          | Certification and Accreditation  |

| Term        | Definition  |
|-------------|---|
| C2          | Command and Control   |
| C3I         | Command, Control, Communications, and Intelligence            |
| C4I         | Command, Control, Communications, Computers, and Intelligence |
| CA          | Customer Advocate   |
| CA          | Certificate Authority   |
| CAC         | Common Access Card  |
| CAN         | Campus Area Network   |
| CAPI        | Cryptographic Application Program Interface                   |
| CBT         | Computer Based Training                                       |
| CCA         | Claimant/Command Agreement                                    |
| CCA         | Clinger-Cohen Act   |
| CCSD        | Command Communications Service Designator (DISA)              |
| CDR         | Critical Design Review  |
| CEP         | Command Execution Plan  |
| CFE         | Contractor Furnished Equipment                                |
| CFO         | Chief Financial Officer                                       |
| CGI         | Common Gateway Interface                                      |
| CINCLANT    | Commander in Chief Atlantic                                   |
| CINCLANTFLT | Commander in Chief, U.S. Atlantic Fleet                       |
| CINCPAC     | Command in Chief Pacific                                      |
| CIO         | Chief Information Officer                                     |
| CLIN        | Contract Line Item Number                                     |
| CM          | Configuration Management                                      |
| CMC         | Commander of the Marine Corps                                 |
| CME         | Command Mission Equipment                                     |
| CMS         | Cryptologic Material Security                                 |
| CMS         | COMSEC Material System  |
| CMS         | Conference Management Services                                |
| CN          | Common Name   |
| COE         | Common Operating Environment                                  |

| Term            | Definition  |
|-----------------|---|
| COM+            | Common Object Model Plus  |
| COMPUTSEC       | Computer Security   |
| COMSEC          | Communications Security   |
| CONOPS          | Concept of Operations   |
| CONUS           | Continental United States   |
| COO             | Chief Operating Officer   |
| COR             | Contract Officer Representative   |
| CORBA           | Common Object Request Broker Architecture   |
| COTR            | Contracting Officer's Technical Representative  |
| COTS            | Commercial Off the Shelf  |
| CPARS           | Contract Performance Assessment Reports   |
| CPU             | Central Processing Unit   |
| CR              | Completion Report (DISA)  |
| CRA             | Continuing Resolution Authority   |
| CRL             | Certificate Revocation List   |
| CS              | Customer Satisfaction   |
| CSS             | Cascading Style Sheets  |
| CT&E            | Contractor Test & Evaluation  |
| CTF             | Commander Task Force  |
| CTO/Engineering | Chief Technology Officer/Engineering  |
| CTR             | Contract Technical Representative   |
| Customization   | Allows the portal administrator to filter the content based on user roles and to push content to the users. |
| DAA             | Designated Approval Authority   |
| DASN            | Deputy Assistant Secretary of the Navy  |
| DATMS-C         | Data ATM Services-Classified  |
| DATMS-U         | Data ATM Services-Unclassified  |
| DBMS            | Database Management System  |
| DCMS            | Director, COMSEC Material System  |
| DCOM            | Distributed Component Object Model  |
| DCS             | Data Conference Services  |

| Term    | Definition  |
|---------|---|
| DDN     | Defense Data Network  |
| DFAS    | Defense Financial Accounting Service  |
| DFS     | Distributive File System  |
| DHTML   | Dynamic Hyper Text Markup Language  |
| DII     | Defense Information Infrastructure  |
| DIMHRS  | Defense Integrated Military Human Resource System                               |
| DISA    | Defense Information Systems Agency  |
| DISN    | Defense Information Systems Network   |
| DITCO   | Defense Information Technology Contracting Office (DISA)                        |
| DITSCAP | Defense Information Technology Security Certification and Accreditation Process |
| DL      | Distribution Lists  |
| DMS     | Defense Messaging Service   |
| DMZ     | De-Militarized Zone   |
| DN      | Distinguished Name  |
| DNA     | Distributed Internet Architecture   |
| DoD     | Department of Defense   |
| DOM     | Department of Marines   |
| DOM     | Document Object Model   |
| DON     | Department of the Navy  |
| DPM     | Deputy Program Manager  |
| DRM     | Design Reference Manual   |
| DSC     | DISN Service Center (DISA)  |
| DSL     | Digital Subscriber Line   |
| DSN     | Defense Switched Network  |
| DTC     | DISN Transition Contract  |
| DTD     | Document Type Definition  |
| DVS     | DISN Video Systems  |
| EAF     | Enterprise Architecture Framework   |
| EDS     | Electronic Data Systems   |
| EFS     | Encrypted File System   |

| Term                             | Definition  |
|----------------------------------|---|
| EIWG                             | Enterprise IT Strategy and Coordination Working Group   |
| EJB                              | Enterprise Java Beans - A Java API developed by Sun that defines the component architecture for multi-tiered systems. EJBs are the objects in a multi-tiered object-oriented J2EE environment, and enable the developers to focus on actual business architecture as opposed to developing the interfaces between the different components themselves |
| Enterprise Portals               | Enable an enterprise to share information and provide web-based access to applications/services to its employees, customers, and partners.  |
| EO                               | Enterprise Operations   |
| EPOC                             | Electronic Piece of Cheese (Psion EPOC-16 operating System)   |
| ESR                              | Enterprise Service Repository   |
| Extensible Markup Language (XML) | An extension/subset of Standard Graphical Markup Language (SGML) specifically designed for WWW dissemination and display of data. It is an open framework in which developers can develop (and, more importantly, standardize and validate against) a tagged data format.   |
| FAD                              | Financial Accounting Data   |
| FFP                              | Firm-Fixed-Price  |
| FIMS                             | Financial Information Management System   |
| FISC                             | Fleet Industrial Supply Center  |
| FIWC                             | Fleet Information Warfare Center  |
| FMB                              | Financial Management Board  |
| FMOS                             | Flexible Master Operational Servers   |
| FOC                              | Full Operational Capability   |
| FSO                              | Facility Security Officer   |
| FTE                              | Full Time Equivalent  |
| FTS                              | Federal Telecommunications Service  |
| FY                               | Fiscal Year   |
| FYDP                             | Five-Year Defense Plan  |
| GETS                             | Government Emergency Telecommunications Service   |
| GFE                              | Government Furnished Equipment  |
| GFI                              | Government Furnished Information  |
| GFI                              | Government Furnished Inventory  |

| Term                | Definition  |
|---------------------|---|
| GIG                 | Global Information Grid   |
| GMO                 | Government Management Office  |
| GPO                 | Group Policy Objects  |
| GUI                 | Graphical User Interface  |
| HAC S&I             | House Appropriations Committee Survey and Inspection Team   |
| HASC                | House Armed Services Committee  |
| HCI                 | Human Computer Interface  |
| HD                  | Help Desk   |
| HDML                | Hand-Held Device Markup Language  |
| HI                  | Horizontal Integration  |
| Horizontal Portal   | Provides general information, no particular focus, variety of subjects, variety of users , and cuts across business functions and applications. Example: Yahoo. |
| Hotlink Integration | Provides as is access to an existing web-based application through a hyperlink or a list of hyperlinks.   |
| HQMC                | Deputy Chief of Staff Marine Corps  |
| HTML                | HyperText Markup Language   |
| HTTP                | HyperText Transfer Protocol   |
| HTTPS               | HyperText Transfer Protocol Secure  |
| I&A                 | Information and Authentication  |
| I/Q/O               | Inquire/Quote/Order (DISA)  |
| IA                  | Information Assurance (Security)  |
| IATF                | Information Assurance Technical Framework   |
| IATO                | Interim Authority to Operate  |
| IAVA                | Information Assurance Vulnerability Alert   |
| ICD                 | Interface Control Document  |
| ID                  | Identification  |
| ID/IQ               | Indefinite Delivery/Indefinite Quantity   |
| IDO                 | Incentive Determining Official  |
| IDS                 | Intrusion Detection Services  |
| IE                  | Internet Explorer   |

| Term    | Definition   |
|---------|--|
| IEEE    | Institute of Electrical and Electronics Engineers  |
| IETF    | Internet Engineering Task Force  |
| IIOB    | Internet Inter ORB Protocol  |
| ILS     | Integrated Logistic Support  |
| IM      | Information Management   |
| IM/IT   | Information Management/Information Technology  |
| INFOCON | Information Operations Condition   |
| INFOSEC | Information Security   |
| IOC     | Initial Operational Capability   |
| IOT&E   | Interoperability Test and Evaluation   |
| IP      | Internet Protocol  |
| IPG     | Integrative Policy Group   |
| IPR     | In-Progress Review   |
| IPT     | Integrated Process Team  |
| IPT     | Integrated Product Team  |
| IR      | Information Request  |
| IRB     | Incentive Review Board   |
| IRC     | Internet Relay Chat  |
| ISDN    | Integrated Switched Digital Network  |
| ISEA    | In Service Engineering Agent   |
| ISF     | Information Strike Force   |
| ISMO    | Information Systems Management Office  |
| ISO     | International Standards Office   |
| ISSO    | Information Systems Security Officer   |
| IT      | Information Technology   |
| IT-21   | Information Technology for the 21 <sup>st</sup> Century  |
| ITI     | Information Technology Infrastructure  |
| ITIA    | Information Technology Infrastructure Architecture   |
| ITSC    | Information Technology Services Center   |
| ITSG    | Information Technology Standards Guidance  |
| J2EE    | Java 2 Enterprise Edition - Introduced in 1995 by sun microsystems. It is an object-oriented language designed |

| Term                              | Definition   |
|-----------------------------------|--|
|                                   | microsystems. It is an object-oriented language designed for the world wide web, similar to c/c++, in which the source is compiled into 'bytecode', which is then interpreted by run-time environment (known as a java virtual machine) on the host machine. |
| J2ME                              | JAVA 2 Micro Edition   |
| Java                              | A general purpose, high-level, object-oriented, cross-platform programming language developed by Sun Microsystems [not an acronym]   |
| Java Database Connectivity (JDBC) | The Java equivalent to ODBC, which acts as the standard database connection language/method for Java applications.   |
| Java Servlet Pages (JSP)          | JSPs are an extension of servlets that allow web developers to dynamically build web pages.  |
| Java Servlet Pages (JSP)          | Java applets that run on the server side as an alternative to Common Gateway Interface (CGI) applications. JSPs are an extension of servlets which allow web developers to dynamically build web pages.  |
| JDBC                              | Java Database Connectivity   |
| JMS                               | Java Messaging Service   |
| JNDI                              | Java Naming and Directory Interface  |
| JPEG                              | Joint Photographic Expert Group  |
| JSP                               | Java Server Pages  |
| JTA                               | Joint Technical Architecture   |
| JTS                               | Java Transaction Services  |
| JV 2010                           | Joint Vision 2010  |
| KM                                | Knowledge Management   |
| LAN                               | Local Area Network   |
| LANTFLT                           | Atlantic Fleet   |
| LDAP                              | Lightweight Directory Access Protocol  |
| LDAPS                             | Lightweight Directory Access Protocol Secure   |
| LE                                | Lead Engineer  |
| LOA                               | Line of Accounting   |
| LOE                               | Level of Effort  |
| LSA                               | Logistics Support Analysis   |
| MAC                               | Moves, Adds, and Changes   |



| Term         | Definition  |
|--------------|---|
| MAN          | Municipal Area Network  |
| MAN          | Metropolitan Area Network   |
| MARCORSYSCOM | Marine Corps System Command   |
| MARFORLANT   | Marine Forces Atlantic  |
| MARFORPAC    | Marine Forces Pacific   |
| MATCOM       | Material Command  |
| MAW          | Marine Aircraft Wing  |
| MCAGCC       | Marine Corps Air Ground Combat Center   |
| MCCDC        | Marine Corps Combat Development Center  |
| MCEN         | Marine Corps Enterprise Network   |
| MCLB         | Marine Corps Logistics Base   |
| MCRC         | Marine Corps Research Center  |
| MCSC         | Marine Corps Supply Command   |
| MCTN         | Marine Corps Tactical Network   |
| Metadata     | Metadata describes how and when and by whom a particular set of data was collected, and how the data is formatted. Metadata is essential for understanding information stored in data warehouses. |
| MFS          | Multifunction Switches  |
| MILCOM       | Military Communications   |
| MILDEP       | Military Department   |
| MILSATCOMM   | Military Satellite Communications   |
| MIME         | Multipurpose Internet Mail Extensions   |
| MLS          | Multi Level Security  |
| MMS          | Microsoft Metadirection Server  |
| MOA          | Memorandum of Agreement   |
| MS           | Microsoft   |
| MSL          | Multi Security Level  |
| MTS          | Microsoft Transaction Server  |
| MWR          | Morale, Warfare and Recreation  |
| NASCAMP      | Navy Switch and Cable Modernization Program   |
| NAVAIR       | Naval Air Systems Command   |

| Term      | Definition                                       |
|-----------|--|
| NAVICP    | Navy Inventory Control Point                     |
| NAVRESFOR | Commander, Navy Reserve Force                    |
| NAVSEA    | Naval Sea Systems Command                        |
| NAVSUP    | Naval Supply Systems Command                     |
| NCR       | National Capital Region                          |
| NCW       | Network Centric Warfare                          |
| NIPRNET   | Non-Secure Internet Protocol Router Network      |
| NITF      | National Imagery Transmission Format             |
| NMC       | Network Management Center                        |
| NMCI      | Navy Marine Corps Internet                       |
| NMS       | Network Management System                        |
| NNIOC     | Naval Network and Information Operations Command |
| NOC       | Network Operating Center                         |
| NSA       | National Security Agency                         |
| NSIPS     | Naval Standard Integrated Personnel System       |
| NTCSS     | Navy Tactical Combat Support System              |
| NTLC      | Network Transport Logistics Center               |
| NTPG      | NMCI Transition Planning Guide                   |
| NVI       | Naval Virtual Internet                           |
| OACT      | Overarching Action Collaboration Team            |
| OCONUS    | Outside – Continental United States              |
| ODBC      | Open Database Connectivity                       |
| OLA       | Office of Legislative Affairs                    |
| OMG       | Open Management Group                            |
| OPCON     | Operational Control                              |
| OPM       | Office of Personnel Management                   |
| OPNAV N6  | Chief of Naval Operations                        |
| OPSEC     | Operations Security                              |
| OpSS      | Open Source Site                                 |
| ORB       | Object Request Broker                            |
| ORN       | Order Request Number                             |

| Term                     | Definition  |
|--------------------------|---|
| OS                       | Operating System  |
| OSD                      | President's, Office of the Secretary of Defense   |
| OT&E                     | Operational Test & Evaluation   |
| OU                       | Operational Unit  |
| OWAN                     | Okinawa Wide Area Network   |
| P2P                      | Point-to-Point  |
| PACOM                    | Pacific Command   |
| PCO                      | Program Contracting Officer   |
| PD                       | Program Directorate   |
| PDA                      | Personal Digital Assistant  |
| PDC                      | Program Designator Code   |
| PDF                      | Portable Document Format  |
| PDR                      | Preliminary Design Review   |
| PEO-IT                   | Program Executive Office for Information Technology   |
| Personalization          | Allows user to change the look of their portal at multiple levels to suit their preferences.            |
| PF                       | Public Folders  |
| PITN                     | Primary Information Transfer Node   |
| PKI                      | Public Key Infrastructure   |
| PKI                      | Public Key Infrastructure   |
| PMO                      | Program Management Office   |
| PMW                      | Program Management Warfare  |
| PO                       | Purchase Order  |
| POAM                     | Plan of Action and Milestones   |
| POC                      | Point of Contact  |
| POM                      | Program Objective Memorandum  |
| POP                      | Point of Presence   |
| PPI                      | Past Performance Information  |
| PPP                      | Priority Placement Program  |
| PR                       | Purchase Request  |
| Presentation Integration | Provides "as-is" access to an already web-enabled application. Requires that the application content be |

| Term   | Definition   |
|--------|--|
|        | rendered within a panel.                                     |
| PRI    | Portal Repository Interface                                  |
| PSTN   | Public Switched Telephone Network                            |
| PVP    | Permanent Virtual Path                                       |
| RA     | Recovery Agent   |
| RBA    | Revolution in Business Affairs                               |
| RCP    | Request for Contractual Procurement                          |
| RD&A   | Research Development and Acquisition                         |
| RDBMS  | Relational Database Management System                        |
| RDF    | Resource Description Framework                               |
| REA    | Request for Equitable Adjustment                             |
| RFP    | Request For Proposal   |
| RFS    | Request for Service (DITCO)                                  |
| RGC    | Routing Group Connector                                      |
| RGCS   | Routing Group Connector Services                             |
| RII    | Reinvestment In Infrastructure                               |
| RMA    | Revolution in Military Affairs                               |
| RMI    | Remote Method Invocation                                     |
| RMS    | Remote Module Server   |
| RPC    | Remote Procedure Calls                                       |
| RSA    | Rivest, Shamir, & Adleman (public key encryption technology) |
| RSS    | (RDF Rich) Site Summary                                      |
| S      | Secret   |
| S/MIME | Secure/Multipurpose Internet Mail Extensions                 |
| SABI   | Secret and Below Interoperability                            |
| SAM    | Status of Acquisition Message (DISA)                         |
| SAN    | Storage Area Network   |
| SAR    | Status of Acquisition Report                                 |
| SAT    | Special Action Team  |
| SBU    | Sensitive But Unclassified                                   |

| Term                | Definition                                |
|---------------------|---|
| SCI                 | Sensitive Compartmental Information       |
| SDP                 | Service Delivery Point (DISA)             |
| SDU                 | Secure Domain Unit (DISA)                 |
| SF                  | Server Farm                               |
| SGML                | Standard Graphical Markup Language        |
| SGML                | Standard Generalized Markup Language      |
| SIMAN               | Station Iwakuni Metropolitan Area Network |
| SIPRNET             | Secure Internet Protocol Router Network   |
| SLA                 | Service Level Agreement                   |
| SME                 | Subject Matter Expert                     |
| SMTP                | Simple Message Transfer Protocol          |
| SOAP                | Simple Object Access Protocol             |
| SOC                 | Security Operations Center                |
| SONET               | Synchronous Optical Network               |
| SOO                 | Statement of Objectives                   |
| SOW                 | Statement of Work                         |
| SP                  | Service Pack                              |
| SPAWAR              | Space and Naval Warfare Systems Command   |
| SPAWARSYSCEN        | Space and Naval Warfare Systems Center    |
| Specialized Portals | Portals specific to a type of business.   |
| SQL                 | Structured Query Language                 |
| SSA                 | Software Support Activity                 |
| SSAA                | System Security Authorization Agreement   |
| SSC                 | SPAWAR Systems Center                     |
| SSC-SD              | SPAWAR Systems Center – San Diego         |
| SSL                 | Secure Sockets Layer                      |
| ST&E                | Security Test & Evaluation                |
| STAT                | Systems Transition Analysis Team          |
| T&E                 | Test and Evaluation                       |
| TCP                 | Transmission Control Protocol             |
| TELCO               | Telephone Company                         |

| Term            | Definition   |
|-----------------|--|
| TFW             | Task Force W   |
| TO              | Task Order   |
| TOD             | Technical Objective Document   |
| TR              | Telecommunications Request (DISA)  |
| TS              | Top Secret   |
| TSO             | Telecommunications Service Order   |
| TSR             | Telecommunications Service Request   |
| UC              | Unclassified   |
| UIC             | Unit Identification Code   |
| URI             | Universal Resource Identifier  |
| URL             | Universal Record Locator   |
| USMC            | United States Marine Corps   |
| USN             | United States Navy   |
| VBA             | Visual Basic for Applications  |
| VBNS            | Very high Bandwidth Network Servers  |
| VCNO            | Vice Chief of Naval Operations   |
| VERA            | Voluntary Early Retirement Authority   |
| Vertical Portal | Service centric portal built around a single service offering or application or business function like ERP, SCM, CRM applications. |
| VIG             | Virtual Interest Group   |
| VPN             | Virtual Private Network  |
| VTC             | Video Teleconference   |
| W3C             | World Wide Web Consortium  |
| WAN             | Wide Area Network  |
| WAP             | Wireless Application Protocol  |
| WEN             | Web Enabled Navy   |
| WIPT            | Working Integrated Product Team  |
| WML             | Wireless Markup Language   |
| WR              | Working Request  |
| XML             | eXtensible Markup Language   |
| XSL             | Extensible Stylesheet Language   |

